DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y LAS COMUNICACIONES

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIÓN

SISTEMAS Y REDES DE TELECOMUNICACIONES

PROYECTO FINAL DE CARRERA

Fusión de Datos para Aplicaciones de Seguimiento en Redes de Sensores

AUTOR: AMELIA PERALES EGEA

TUTOR: SARA PINO POVEDANO

8 de mayo de 2013

TÍTULO: *Fusión de Datos para Aplicaciones de Seguimiento en Redes de Sensores.*

AUTOR: *AMELIA PERALES EGEA*

TUTOR: *SARA PINO POVEDANO*

La defensa del presente Proyecto Fin de Carrera se realizó el día 8 de mayo de 2013; siendo calificada por el siguiente tribunal:

PRESIDENTE: *Matilde Sánchez Fernández*

SECRETARIO *Katrin Achutegui Roncal*

VOCAL *José María Sierra Cámara*

Habiendo obtenido la siguiente calificación:

CALIFICACIÓN:

Presidente **Secretario** **Vocal**

A mi tío Gonzalo.

A mi abuelo Pelos.

A Alejandra.

Agradecimientos

Quisiera agradecer a todas aquellas personas que confiaron en mí, que me dieron fuerzas para continuar cuando creía firmemente que todo estaba perdido, me tendieron la mano en momentos difíciles y me ayudaron a levantarme.

A mi tutora Sara por darme la oportunidad de realizar este Proyecto de Fin de Carrera, por su inestimable ayuda, por tener paciencia infinita conmigo; por ser amiga y confidente.

A mi familia. A mi gran familia... A mis padres, Amelia y Juan, por su amor incondicional, sus consejos y su comprensión; por enseñarme que con constancia y esfuerzo, pasito a pasito, se puede alcanzar la luna. A mi hermana Beatriz, por ser hermana y amiga, por su calidez y su fuerza; por estar ahí cuando lo necesito. A mis abuelos, a mis tíos, a mis primos, a mi cuñado Sergio... no hay nada mejor que sentirse parte de vosotros.

A Paco, por enseñarme que la pasión y el entusiasmo mueven el mundo; por escucharme y saber hacerme reír a carcajadas aún cuando parece imposible. Como dice la canción: 'tú me haces bailar...'

A todos mis amigos. A mis amigas de toda una vida, a los que me acompañan desde el colegio, a los que hice en la universidad, los que encontré en Finlandia y los que me arroparon a mi vuelta... gracias por dejarme formar parte de vuestro pequeño gran universo; porque sabéis cómo soy y me aceptáis sin condición alguna.

Sin unos ojos que lo lean,
un libro contiene signos que no producen conceptos.
Y por lo tanto, es mudo.
Guillermo de Baskerville
(En el nombre de la rosa - Umberto Eco)

A veces me convenzo de que la estupidez se llama triángulo,
de que ocho por ocho es la locura o un perro.
Rayuela (Julio Cortázar)

Resumen

Las redes de sensores inalámbricos producen gran cantidad de datos que necesitan ser procesados, enviados y evaluados para conseguir un objetivo concreto, como localizar un blanco en movimiento. La forma que estos datos son manipulados por los nodos de la red es una cuestión fundamental, pues el consumo de energía u otros recursos difiere en gran medida del tratamiento de la información en cada paso. Las técnicas de fusión de datos combinan de forma óptima fuentes de información redundantes para reducir el tráfico presente en la red y permitir así un aumento de la vida útil del sistema.

El desarrollo de este Proyecto de Fin de Carrera se basa en la aplicación de filtros de partículas distribuidos para la resolución del problema de estimación de la posición de un objeto en movimiento dentro de una red de sensores.

A lo largo de este estudio se presentan un conjunto de soluciones basadas en lo anteriormente expuesto capaces de calcular los parámetros correspondientes para obtener una estimación de la posición de un blanco a lo largo del tiempo y proporcionar unas prestaciones de funcionamiento adecuadas.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

INTRODUCCIÓN

1.1. Motivación

A lo largo de los últimos años se ha despertado el interés en las Redes de Sensores Inalámbricos *(Wireless Sensor Networks)* tanto por las características que posee este tipo de sistemas como por el desarrollo de numerosas aplicaciones. La gran cantidad de información que pueden recoger y almacenar o procesar hace que sean aplicables en numerosos campos y proporcionen mejoras a sistemas médicos, medioambientales, industriales o militares, entre otros.

Uno de los rasgos comunes a muchas de las aplicaciones desarrolladas para este tipo de redes es la detección de eventos, entre los que se encuentra el seguimiento espacio-temporal de un móvil dentro del área de cobertura de la red a través de la estimación óptima de su estado, aplicación en la que se centra este Proyecto de Fin de Carrera.

Para conseguir la resolución del problema de detección y seguimiento que se plantea se pueden utilizar numerosas técnicas. Este estudio se centra en la aplicación del filtrado de partículas. La idea es determinar una serie de parámetros que describan un sistema que evoluciona en el tiempo, a través de una aproximación numérica que plantea una solución efectiva a dicho problema estadístico.

Aunque hace unos años el desarrollo de este tipo de filtros en aplicaciones reales era limitado debido a su complejidad de cálculo y a la limitación computacional de los recursos disponibles,

los grandes avances en computación de la última década y el destacado potencial de esta metodología la han posicionado como un área emergente de investigación y desarrollo.

Una de las desventajas principales de estos sistemas es el gran gasto de energía que limita la vida útil de la red. Las técnicas de fusión de datos se presentan como una solución a este problema buscando combinar de forma óptima fuentes de información redundantes. Aplicándolas en el algoritmo de enrutamiento en lugar de en el nodo destino permite reducir el tráfico presente en la red, lo que conlleva un menor gasto de energía y un aumento de la vida de cada nodo que forma la red.

A lo largo de este Proyecto de Fin de Carrera se presentan un conjunto de soluciones basadas en algoritmos distribuidos de filtros de partículas en redes de sensores inalámbricos sobre las que se aplican técnicas de fusión de datos, analizando dichas soluciones para determinar su comportamiento y eficacia tanto en términos de error y retardo en la estimación, como de tiempo de vida de la red.

1.2. Objetivos

Al inicio de este Proyecto de Fin de Carrera se marcaron una serie de propósitos que se resumen a continuación. El *objetivo principal* es presentar una serie de soluciones al problema de detección y seguimiento de un blanco en redes de sensores utilizando para ello algoritmos de filtros de partículas. Se deberá además analizar, implementar y comparar a través de simulaciones las soluciones propuestas.

Como objetivos secundarios se pueden enumerar los siguientes:

- Realizar un análisis general de las redes de sensores inalámbricas, para conocer sus características y funcionalidades.
- Estudiar los filtros de partículas como aproximación numérica a la distribución de probabilidad a posteriori que posibilitará la estimación de la posición del blanco a lo largo del tiempo.
- Estudiar los filtros de partículas distribuidos y determinar un modelo general para su implementación en este proyecto, además de posibles variantes que puedan ayudar a mejorar

el algoritmo principal.

- Analizar las principales técnicas de asociación de datos, agregación y fusión de datos, y plantear un sistema de fusión de datos que sea aplicable en los algoritmos anteriores.

1.3. Organización del PFC

A lo largo de los capítulos que conforman este Proyecto de Fin de Carrera estudiaremos la aplicación de diversos algoritmos distribuidos de filtros de partículas en sistemas de redes de sensores inalámbricos para su aplicación en problemas de detección y seguimiento de blancos en movimiento.

En este primer capítulo se intenta situar al lector en el contexto del proyecto, explicando en qué consiste, sus motivaciones principales y los objetivos que pretenden cumplirse a lo largo de su desarrollo.

En el capítulo siguiente *(Capítulo 2)* se introducen nociones básicas sobre las redes de sensores inalámbricas y sus características, enumerando posibles campos de aplicación, así como la descripción de un escenario típico de una red de este tipo para la detección y seguimiento de blancos en movimiento.

En el *Capítulo 3* se muestra el modelo de sistema. Se presenta el modelo dinámico que caracterizará el movimiento del blanco, del sensor utilizado, de la observación obtenida por este y de la arquitectura del sistema.

En un cuarto capítulo *(Capítulo 4)* se desarrolla la formulación necesaria para el seguimiento de blancos centrándonos en los filtros de partículas como principal algoritmo de seguimiento.

En el *Capítulo 5* se muestran las variantes propuestas para los filtros de partículas que pretenden cumplir los objetivos marcados en la sección anterior.

En el sexto capítulo *(Capítulo 6)* se describen las simulaciones realizadas de cada uno de los modelos propuestos, analizando sus resultados y llegando a una serie de conclusiones en el

Capítulo 7 en base a estos últimos y, proponiendo ciertas líneas de trabajo futuras.

Capítulo 2

REDES DE SENSORES INALÁMBRICOS

En la última década los sistemas basados en redes de sensores inalámbricos han experimentado un gran crecimiento, aplicándose en ámbitos tan diversos como la sismología, la botánica, el control del tráfico urbano, la ecología o la medicina.

A lo largo de este capítulo se presenta una definición de las redes de sensores inalámbricos, sus características principales y alguna de las aplicaciones desarrolladas. Así mismo se introduce su utilización en el problema de detección y seguimiento de blancos en movimiento con el que trabajaremos a lo largo de este estudio, describiendo un escenario típico de este tipo de sistemas.

2.1. Redes de sensores

Se puede definir una red de sensores como un grafo donde los nodos son sensores y las ramas representan los enlaces de interconexión. En el caso de una red de sensores inalámbricos este enlace es una conexión directa, de un solo salto, siendo los vecinos de un sensor concreto aquellos que se encuentran dentro de su rango de alcance [2].

Estas redes de sensores se caracterizan principalmente por estar formadas por un gran número de nodos densamente desplegados que colaboran en una tarea común. Ya que tanto el tamaño como el coste de los nodos se ha visto reducido en gran medida en los últimos años, estas redes son capaces de, con pocos recursos, medir parámetros, almacenarlos, procesarlos y enviarlos proporcionando información útil al usuario final localizada en tiempo y/o espacio según las ne-

cesidades del problema a tratar. En la Figura 2.1 se muestra un ejemplo de sensor inalámbrico, en el que se puede observar su reducido tamaño.

Figura 2.1: Sensor inalámbrico

En este documento nos centraremos únicamente en redes de sensores inalámbricos. Los principales componentes de estos dispositivos inteligentes pueden observarse en la Figura 2.2 y son el transceptor, la fuente de alimentación, el microcontrolador, la memoria externa y el sensor o sensores:

- **Transmisor:** Es el componente que hace posible la comunicación de los nodos con sus vecinos, permitiendo el intercambio de información para así llevar a cabo el procesamiento colaborativo de las señales. En redes inalámbricas de sensores, los transistores suelen estar basados en el estándar IEEE 802.15.4, versión reducida del estándar de comunicaciones Wifi [5].

- **Fuente de alimentación:** Proporciona energía a las distintas tareas que el sensor realiza, como el sensado, la comunicación entre nodos o el tratamiento de la información, siendo la comunicación aquella que necesita mayor cantidad de energía. Debido al tamaño de los nodos no resulta posible utilizar grandes fuentes de energía, lo que constituye una de las grandes limitaciones de este tipo de redes.

- **Microcontrolador:** Realiza el procesamiento de datos así como tareas de control sobre el funcionamiento de otros bloques del nodo. Se caracterizan por su bajo coste, su flexibilidad para conectarse con otros dispositivos, su facilidad de programación y su bajo consumo de energía.

- **Memoria externa:** Almacenan parte de la información procesada. Se utilizan dispositivos que consuman pocos recursos por las limitaciones energéticas que caracterizan este tipo de redes. Desde un punto de vista energético las memorias externas más relevantes que suelen

encontrarse en estos dispositivos son las memorias Flash. Estas se utilizan dado su bajo coste frente a su capacidad de almacenamiento.

- **Sensor:** Son aquellos dispositivos hardware que nos proporcionan las muestras de información sobre cambios en el entorno. Obtienen datos físicos sobre el parámetro a monitorizar y digitalizan la señal analógica recibida a través de un conversor A/D para posibilitar así el procesado de la información.

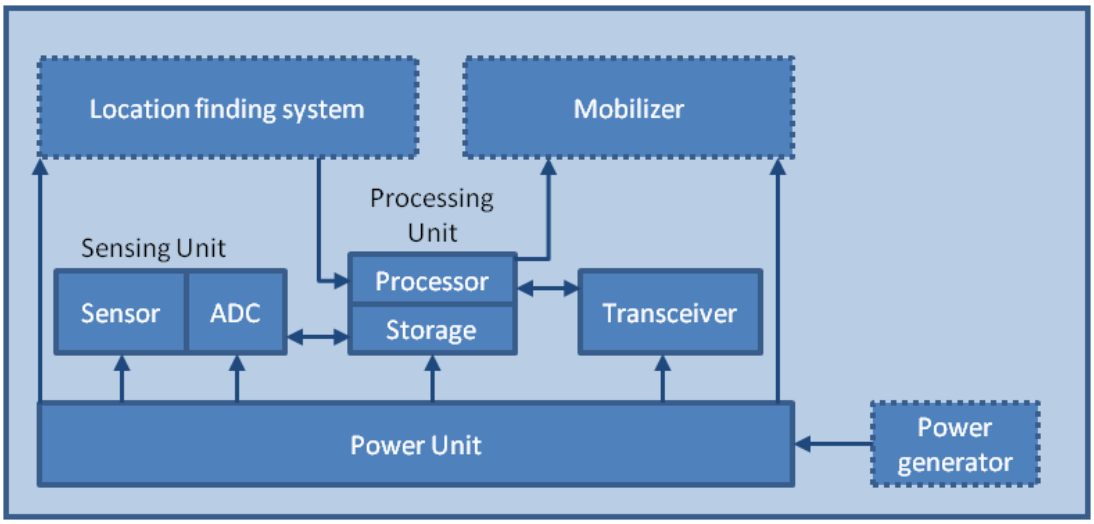


Figura 2.2: Esquema de un Nodo en la Red de Sensores [1]

2.2. Características de las redes de sensores

Las redes de sensores tienen una serie de características propias y otras adaptadas de las redes Ad-Hoc. En esta sección hablaremos de ellas, determinando tanto sus ventajas como sus limitaciones. Todas ellas han de tenerse en cuenta en el desarrollo de nuevas aplicaciones o tecnologías basadas en redes de sensores inalámbricos:

- **Topología dinámica:** La topología de una red de sensores inalámbricos es habitualmente cambiante; los nodos pueden modificar su localización después del despliegue inicial debido a fenómenos medioambientales como el viento o el agua o puede ser una característica deseada en el sistema por el tipo de aplicación implementada. Sea cual fuere la razón, en ciertas aplicaciones los nodos deben poseer capacidades de adaptación para poder comunicar los datos adquiridos. El grado de movilidad puede variar de movimientos ocasionales

entre de grandes periodos de inmovilidad, o puede ser constante. A la hora de implementar una red de sensores inalámbricos con estas características se ha de estimar la velocidad de movimiento del sensor, pues esta tiene gran relación con el periodo de tiempo durante el cual los nodos pueden estar fuera del rango de comunicación de sus vecinos.

- **Tolerancia a errores:** Un dispositivo sensor dentro de una red sensora tiene que ser capaz de seguir funcionando a pesar de tener errores en el sistema propio.

- **Consumo energético:** Dependiendo de la aplicación, la vida de una red de sensores puede variar en un rango muy amplio, desde algunas horas hasta varios años. La vida útil del sistema es así de gran importancia pues determina el grado de eficiencia energética la resistencia del nodo. Debido a que son dispositivos autónomos con un tamaño limitado su unidad de energía está también limitada. Aunque la capacidad energética puede variar según el coste del nodo o el dispositivo energético utilizado (tanto baterías como células solares, por ejemplo), un nodo sensor tiene que contar con un procesador de consumo ultra bajo así como de un transceptor radio con la misma característica. Su software ha de conjugar también dicha propiedad haciendo el consumo aún más restrictivo, limitando por tanto su complejidad.

- **Algoritmos descentralizados:** La mayoría de sistemas de sensado tiene una naturaleza descentralizada que los hacen más robustos frente a un fallo en un enlace por la redundancia presente en la red. Su gran escalabilidad radica en esta descentralización, siendo la única opción viable para conseguir el tamaño suficiente que algunas aplicaciones demandan.

- **Configuración multisalto:** El canal radio es un canal muy variable en el que existen una serie de fenómenos que pueden producir errores en los datos, como pueden ser la atenuación, los desvanecimientos rápidos, los desvanecimientos lentos y las interferencias. Una configuración multisalto proporciona un ahorro de energía significante frente a una red de salto único desplegada en un área similar.

  Tomando como ejemplo una red básica con N saltos y asumiendo que la distancia entre nodos es constante y de valor r, la distancia total de la red a cubrir será Nr [2]. Si la potencia mínima recibida en un nodo para una tasa de error dada es P_r, y la transmitida por el nodo transmisor es P_t podemos determinar la atenuación por radiofrecuencia en un salto a través de la ecuación (2.1) donde α es el factor de atenuación de radiofrecuencia

que tiene en cuenta los desvanecimientos por multitrayecto y otras interferencias.

$$P_r \propto \frac{P_t}{r^\alpha} \Rightarrow P_t \propto r^\alpha P_r \tag{2.1}$$

Por tanto, la ventaja de transmisión de una configuración multisalto frente a otra de salto único vendrá dada por la ecuación (2.2) y puede observarse en la Figura 2.3. Considerando únicamente el efecto de transmisión en radiofrecuencia a mayor número de saltos mayor ahorro de energía. A efectos reales existen componentes circuitales que consumen energía con cada transmisión o recepción, por lo que un diseño óptimo deberá llegar a un compromiso entre el consumo de energía total y la eficiencia obtenida usando este método de enrutamiento.

$$\eta_{RF} = \frac{P_{t[Nr]}}{NP_{t[r]}} = \frac{(Nr)^\alpha P_r}{N \cdot r^\alpha P_r} = N^{\alpha-1} \tag{2.2}$$

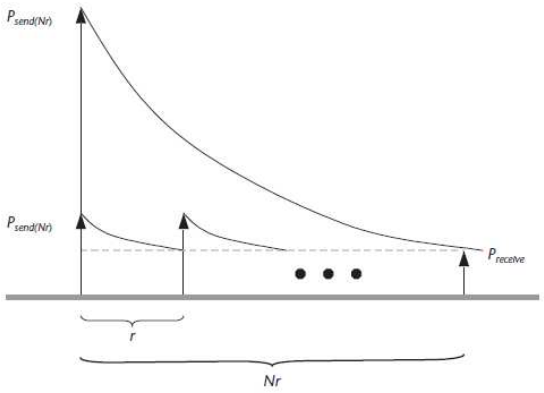


Figura 2.3: Ventaja energética de una configuración multisalto frente a otra de salto único [2]

- **Costes de producción:** Las redes de sensores están normalmente formadas por un número muy elevado de nodos. El coste de su fabricación es reducido, una vez que han sido adaptados a una aplicación determinada, al ser fabricados en grandes cantidades.

- **Limitación de Hardware:** Cada nodo tiene una capacidad limitada de procesamiento, almacenaje y comunicación, así como de autosuministro de energía. Para poder conseguir un menor consumo de potencia el hardware ha de ser lo más sencillo posible; sencillez que ha de aplicarse también a su transceptor radio.

- **Limitación en el soporte de red:** Debido a la naturaleza de las aplicaciones que este tipo de redes tiene, normalmente no se puede realizar un soporte físico de la red. La topología de la red es por lo tanto dinámica y variable, pues los nodos pueden dejar de funcionar de forma paulatina según la carga de procesamiento y comunicación de datos que hayan llevado. Esta limitación resulta importante a la hora de realizar la configuración o el enrutamiento.

- **Limitación en el desarrollo del software:** Además de las limitaciones en la complejidad del software por la necesidad de reducción del coste energético hay que tener en cuenta que las tareas a realizar son normalmente en tiempo real y distribuidas, lo que hace necesario la colaboración entre nodos teniendo que manejar varios eventos al mismo tiempo que compiten entre sí.

- **Eficiencia en detección:** Ya que el rango de sensado de un nodo está directamente relacionado con el nivel de potencia de este dispositivo, una red densamente implementada mejorará las oportunidades de detección de un blanco dentro de ese rango; el incremento del número de nodos en una misma área reduce la distancia entre nodos mejorando la relación señal a ruido, SNR, del nodo.

  Considerando una red de sensores en la que la distancia entre nodos viene dada por r, siendo $\alpha = 2$ el factor de atenuación en radiofrecuencia en este caso, la potencia recibida viene dada por la expresión (2.3) y su relación señal a ruido está definida como (2.4), donde P_n es la potencia de ruido existente en el dispositivo.

$$P_r \propto \frac{P_t}{r^\alpha} \tag{2.3}$$

$$\begin{aligned} SNR_r &= 10 \log \frac{P_r}{P_n} \\ &= 10 \log \frac{P_t}{r^2 P_n} \\ &= 10 \log P_r - 10 \log P_n - 20 \log r \end{aligned} \tag{2.4}$$

  Aumentando la cantidad de nodos existentes en la red en un factor K se reduce la distancia entre nodos en un factor $\frac{1}{\sqrt{K}}$, lo que provoca un incremento de la SNR del sensor de $10logK$

dBs, como puede verse en la ecuación (2.5).

$$\begin{aligned} \eta_{SNR} &= SNR_{\frac{r}{\sqrt{K}}} - SNR_r \\ &= 20 \log \frac{r}{\frac{r}{\sqrt{K}}} \\ &= 10 \log K \end{aligned} \tag{2.5}$$

Tras exponer las principales características de las redes de sensores se puede concluir que la transmisión y recepción de medidas entre nodos conlleva un gasto de energía muy alto. Al estar formadas por dispositivos autónomos operan con unas limitaciones importantes de computación y comunicación impuestas precisamente por esta característica de autonomía de los nodos, ya que son dispositivos con una batería limitada[6]. En muchas de sus aplicaciones (medioambientales o militares) acceder periódicamente a los nodos de la red para recargar o reemplazar sus baterías resulta del todo imposible. Debido a esta restricción el objetivo principal en el desarrollo de nuevas técnicas ha de ser alargar la vida útil de los sensores y con ello la de la red.

Teniendo en cuenta que son los procesos de comunicación entre nodos los que normalmente consumen más energía, la arquitectura de la red tendrá una gran influencia en la eficiencia energética de los procesos de seguimiento de blancos. La configuración más convencional a este respecto consiste en una red centralizada en la que los sensores recolectan información y la envían a un nodo destino (*sink node*), encargado de realizar todo el procesamiento de datos requerido para la localización y seguimiento del blanco. Aunque resulta una ventaja disponer de toda la información relevante posible para la estimación de la trayectoria, el tráfico generado en la transmisión de la información hace que la batería de los sensores se gaste de forma prematura [6].

Existen muchas estrategias diferentes para conseguir un ahorro en el gasto de energía; en este estudio nos centraremos en algunas de ellas. Se intenta optimizar la implementación de los nodos bien reduciendo el tráfico de datos en la red bien reduciendo el número de sensores envueltos en el seguimiento de cada uno de los blancos.

2.3. Aplicaciones

Los avances que en la última década se han producido tanto en el diseño e implementación de redes inalámbricas como en el desarrollo de los sensores (pequeños, baratos y autónomos) han permitido la aparición de redes de sensores a gran escala perfectamente capacitadas para su utilización en muy diversas aplicaciones comerciales y militares actualmente de gran interés. Podemos encontrar algunos ejemplos a continuación:

- **Conservación medioambiental:** Este tipo de aplicaciones están caracterizadas por la durabilidad de los sensores en un entorno inaccesible a lo largo de un largo periodo de tiempo. Algunas de ellas ellas son:
  - Seguimiento y documentación de hábitats salvajes de forma poco invasiva.
  - Seguimiento de animales en libertad.
  - Detección de focos en incendios forestales para disminuir el tiempo de respuesta de los servicios de emergencia.
  - Detección de inundaciones.
  - Estudios de contaminación.
  - Prevención de desastres y/o monitorización de áreas afectadas.
  - Estudios sísmicos y volcánicos (Figura 2.4)

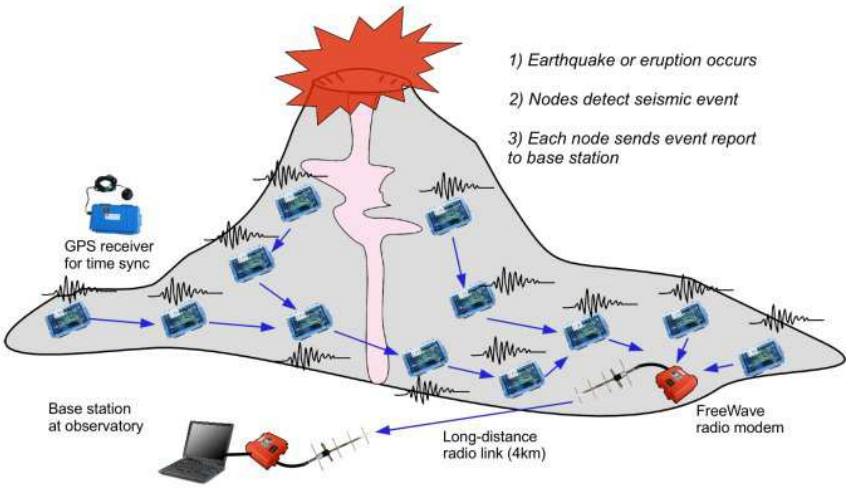


Figura 2.4: Detección de erupciones mediante Redes de Sensores inalámbricas [3]

- **Zonas urbanas:** La monitorización y el seguimiento hace posible una reducción significativa del coste tanto del servicio como del mantenimiento, incrementando el tiempo de vida útil de los equipos, mejorando la satisfacción del cliente e incluso salvando vidas. Dentro de estas aplicaciones podemos encontrar:
  - Monitorización de tráfico en carreteras.
  - Seguimiento de cadenas de montaje en fábricas inteligentes.
  - Control y monitorización del consumo de electicidad.
  - Seguimiento y vigilancia en aeropuertos.
- **Aplicaciones militares:** Alguna de las características más importantes que este tipo de redes de sensores tienen es la habilidad de resistir todo tipo de impactos y explosiones, además de la interoperabilidad con otros sistemas existentes.
  - Seguimiento de equipos enemigos.
  - Detección de cruces ilegales de frontera.
  - Detección de ataques biológicos, químicos o nucleares.
  - Protección del perímetro de una base militar en ambientes hostiles.
- **Aplicaciones médicas:** Comparados con el uso de técnicas convencionales, las soluciones en este campo basadas en redes de sensores inalámbricos demuestran ser menos invasivas para el paciente mejorando al mismo tiempo la precisión de los datos medidos.
  - Telemonitorización de datos fisiológicos en pacientes.
  - Administración de medicamentos.
  - Seguimiento de médicos y pacientes en hospitales.

Hemos de tener en cuenta que hoy en día el estudio y desarrollo de redes de sensores supone un área de investigación y desarrollo multidisciplinar, pues este está directamente ligado al procesado de señales, las bases de datos y su gestión, encaminamiento en redes y protocolos, los algoritmos distribuidos o la arquitectura de sistemas [2]. Debido a la gran cantidad de áreas en las que se muestran útiles continuamente aparecen nuevas aplicaciones que necesitan redes de sensores con mejores prestaciones que las presentes. Es necesario por tanto desarrollar

nuevos algoritmos y protocolos y diseñar nuevas metodologías y herramientas para soportar el procesamiento de señales distribuidas, así como mejorar gestión y almacenaje de los datos, desarrollando nuevas topologías de red e incrementando tanto el número de nodos por aplicación como la descentralización del control y procesado de la red, tal y como podemos observar en la Figura 2.5.

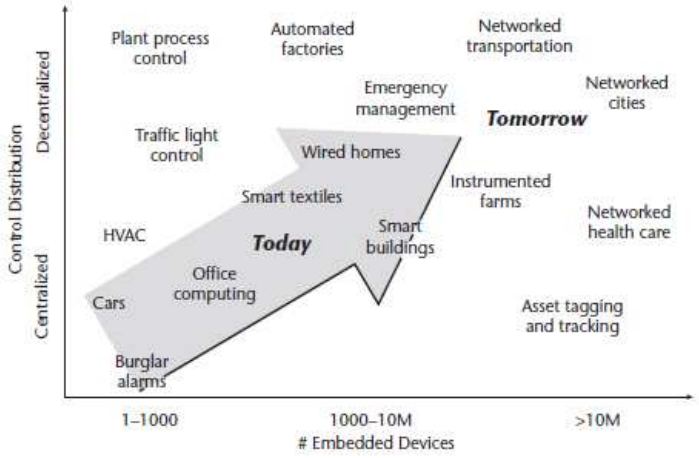


Figura 2.5: Evolución de las Redes de Sensores [2]

2.4. Detección y seguimiento en redes de sensores

Muchas de las aplicaciones de las redes de sensores conllevan la detección y seguimiento de uno o varios objetivos. En un sistema de detección y seguimiento de blancos los objetos de interés se mueven dentro del área de alcance de los sensores. Estos miden la potencia de la señal para hacer posible la estimación tanto de sus trayectorias como de sus velocidades. En este tipo de aplicaciones el sensor adquiere datos, los procesa y los reenvía con el fin de, a través de la colaboración de la red en su conjunto, cumplir con la tarea encomendada.

La localización y el seguimiento de blancos en movimiento es un problema tan común en aplicaciones de este tipo de redes que puede ser usado como forma de estudio sobre el procesado de información o la organización de la red. Por ejemplo, uno de los principales problemas del procesado de señal e información de forma colaborativa *(CSIP - Collaborative signal and infor-*

mation processing) es el establecimiento de forma dinámica de grupos de sensores basado en los requerimientos de las tareas que realizan y la disponibilidad de sus recursos. El seguimiento pone de manifiesto las cuestiones más importantes que rodean el procesado colaborativo: el intercambio de información y la gestión de grupos de nodos incluyendo la cuestión de qué nodos son los que deberían estar activos en cada momento escuchando para obtener información, cuáles poseen información útil y deberían comunicarla a sus vecinos, cuáles deberían recibir la información y cada cuánto tiempo, y todo ello dentro de un marco que evoluciona de forma dinámica [2].

Desde un punto de vista de sensado y procesado de información podemos definir una red de sensores como un conjunto de datos $G =< V, E, P_V, P_E >$ en el que V y E especifican el grafo que forma la red, formado por V nodos y $E \subseteq V\,x\,V$. P_V define el conjunto de funciones que caracterizan cada nodo de V como su localización, su capacidad computacional, su modalidad de sensado o su reserva de energía entre otras características. Así los sensores pueden ser acústicos, magnéticos, sísmicos, de infrarojos, de temperatura o de luz. Y la señal de información que envían a los nodos vecinos puede incluir datos sobre la amplitud de la señal recibida, la dirección de llegada de esa fuente de información, el rango de área en el que puede encontrarse el blanco o etiquetas que clasifiquen los distintos objetos que pueden detectar. De forma similar P_E especifica las propiedades de cada enlace, como su capacidad o su calidad.

En el estudio del seguimiento de blancos en movimiento el procesado de información de forma colaborativa ha de extenderse a lo largo de la línea temporal y del área que la red ocupa. Así el objetivo principal suele ser poder localizar un objeto en un tiempo limitado usando una cantidad de energía también limitada. Esto nos lleva a buscar soluciones que proporcionen una optimización tanto de la energía consumida como de los recursos disponibles.

Hay que tener en cuenta que en este tipo de redes de sensores inalámbricas algunas de las características que definen tanto al blanco como las limitaciones de la red están disponibles únicamente a tiempo real, en el proceso de ejecución. Así, aunque el problema de optimización puede resolverse de formas muy variadas, una de ellas es la utilización de algoritmos que tengan en cuenta esa variación dinámica de las limitaciones de la red y que, por tanto, vayan evolucionando en el tiempo.

2.4.1. Escenario típico

En un sistema de detección y seguimiento los blancos se mueven dentro de un área de vigilancia en la que se encuentran desplegados los sensores. Estos realizan una medida que depende en cierta forma de la posición del blanco, haciendo así posible la estimación tanto de la posición como de la velocidad de los blancos. Para ello los sensores han de recolectar datos, procesar información y comunicarse entre ellos.

De forma general, suponiendo que un blanco X se mueve dentro de una red de sensores de izquierda a derecha, podemos establecer una serie de tareas que se inician entre los nodos de la red y que se observan en la Figura 2.6:

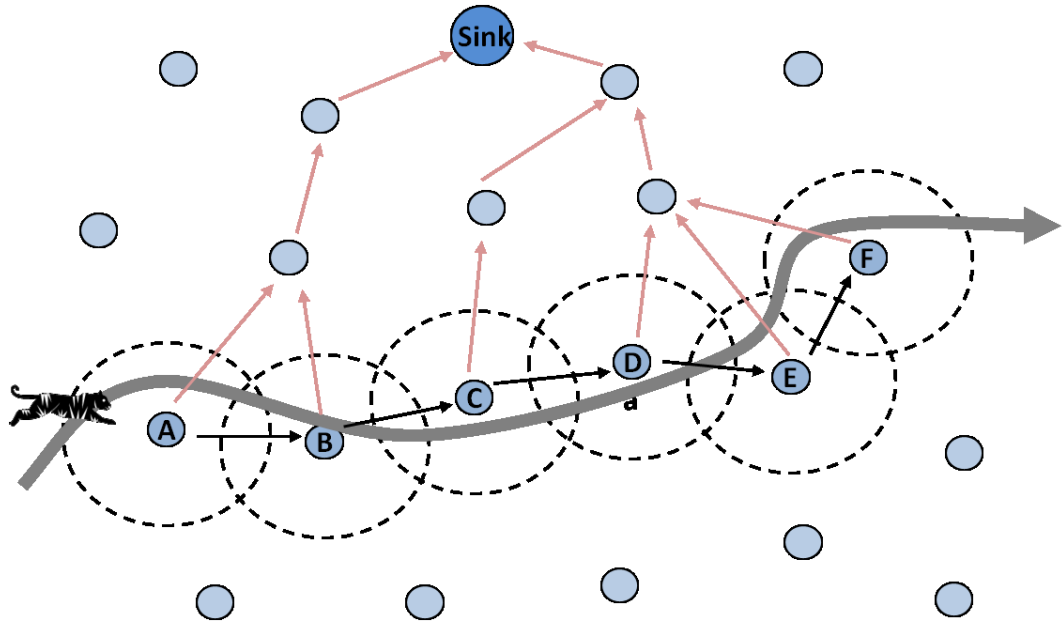


Figura 2.6: Escenario de seguimiento de blancos X e Y, en una red de sensores

- **Detección:** El nodo A detecta el blanco X y envía su medida a través de la red hacia el nodo destino (*sink node*). El mismo procedimiento lo llevan a cabo todos aquellos nodos vecinos que han detectado el blanco dentro de su rango de alcance. Comienza así la localización y el seguimiento del objetivo.
- **Procesado colaborativo en el nodo destino:** El nodo destino, una vez ha recibido las medidas, procesa toda la información para obtener la posición del blanco dentro de la red.

Esta estimación de la posición puede determinarse a través de una triangulación (utilizando las medidas de un conjunto de sensores) o puede realizarse, por ejemplo, usando un método estadístico como la estimación Bayesiana.

- **Comunicación:** Como el blanco X se encuentra en movimiento, el nodo A debe ceder su testigo sobre la localización del objetivo al nodo B una vez que este desaparece de su alcance, y B al nodo C y así sucesivamente. Uno de los principales retos en el diseño de este tipo de redes es la selección del siguiente nodo dentro del conjunto de vecinos del nodo inicial. Una elección equivocada, o no lo suficientemente buena, puede causar una pérdida del objeto o incurrir en una comunicación innecesaria y elevada que limite la vida útil de la red.

- **Fin del seguimiento:** Una vez que el objeto en movimiento desaparece del área de acción de la red, los nodos que la conforman escuchan de forma periódica hasta que detectan otro blanco y el proceso comienza de nuevo.

El escenario de seguimiento de blancos en movimiento descrito en esta sección pone de manifiesto una serie de factores importantes que han de tenerse en cuenta a la hora de realizar el diseño de una red de sensores:

- Dentro del procesado colaborativo se ha de tener en cuenta la detección del objeto, su localización y seguimiento o el control de las tareas de sensado.

- En la transmisión es importante la clasificación de los datos, su agregación o fusión, y su enrutamiento.

- En las bases de datos destaca la abstracción de los datos y la optimización de la búsqueda.

- En el interfaz de usuario la exploración de los datos, la búsqueda y la visualización de los resultados.

- Dentro de los servicios de infraestructura, la inicialización de la red, la seguidad o la gestión de fallos.

2.5. Métodos de asociación de datos

Las redes de sensores inalámbricos producen gran cantidad de datos que necesitan ser procesados, enviados y evaluados para conseguir localizar el blanco en movimiento. La forma que estos datos son malipulados por los nodos de la red es una cuestión fundamental, pues el consumo de energía u otros recursos difiere en gran medida del tratamiento de la información en cada paso. Utilizando métodos de optimización de energía como los que se detallan a continuación se consigue un aumento considerable de la vida útil del sistema. Estos métodos tratarán de reducir la cantidad de tráfico existente en la red, eliminando en algunos casos las medidas por debajo de un umbral de potencia o muy ruidosas o realizando otras predicciones sobre el objeto que está siendo monitorizado.

Algunas estrategias a seguir para ahorrar energía son:

- Optimizar el funcionamiento del sensor.
- Minimizar el número de nodos activos en el proceso de localización y seguimiento.
- Reducir la cantidad de datos a enviar, eliminando la menor cantidad de información posible.

En este estudio aplicaremos estrategias que logran reducir el tráfico presente en la red disminuyendo la cantidad de datos a enviar. Son técnicas que utilizan la correlación de datos (correlación temporal o espacial) para eliminar información redundante e innecesaria. Se apoyan en la capacidad computacional presente en los nodos, reduciendo así el tráfico en la red de forma distribuida. Dentro de ellas se encuentran las técnicas de predicción, de procesamiento en red o de fusión y agregación de datos.

Dependiendo de la organización de la red el procesado de datos se puede llevar a cabo:

- **De forma distribuida:** Si todos los nodos de la red realizan las mismas funciones.
- **De forma jerárquica:** Si existen ciertos nodos dentro de la red que realizan funciones distintas a los restantes. Así algunas técnicas dividen la red en grupos de nodos o *clusters* siendo los llamados *cluster-heads*, nodos centrales de estos grupos, los encargados de realizar tareas de procesado de información y de iniciación del enrutamiento [6].

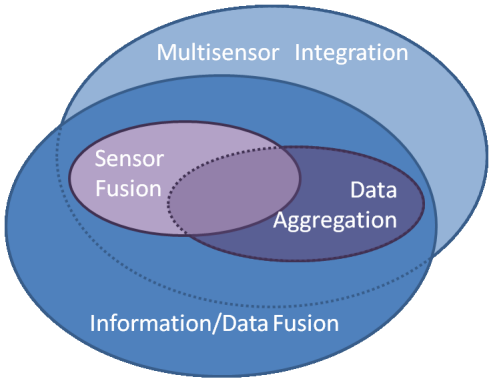


Figura 2.7: Relación entre los métodos de optimización de energía: agregación y fusión de datos [7]

A continuación se muestran dos de los principales métodos de optimización que han sido estudiados: la agregación y fusión de datos.

2.5.1. Agregación de datos

Los métodos de agregación de datos se basan en la agrupación de los datos en paquetes de tamaño determinado, enviándolos al nodo central a través de la red, siendo este último el que realiza el procesado de toda la información recibida.

En la Figura 2.8 se puede observar que cuando dos paquetes llegan a un mismo nodo este agrupa la información recibida en un único paquete sin procesarla, enviando el mensaje resultante hacia el destino. Sin embargo, aunque el tráfico existente en la red disminuye, aumenta el tamaño de los paquetes que viajan a través de esta. Suponiendo que los dos recibidos tengan un tamaño similar, el paquete resultante de la aplicación de esta técnica tendrá prácticamente el doble del tamaño de cualquiera de ellos, restando únicamente el tamaño de una de las cabeceras.

En aquellas aplicaciones en las que se apliquen métodos de agregación de datos resulta importante determinar el tamaño óptimo del mensaje. Se ha de tener en cuenta para ello que:

- Si el tamaño del paquete es demasiado pequeño el tráfico presente en la red sigue siendo demasiado y los problemas existentes en el esquema inicial no se ven reducidos.
- Si el tamaño del paquete es demasiado grande los nodos intermedios tendrán que esperar

más tiempo a que lleguen una gran cantidad de medidas. El retraso de estas medidas al nodo destino aumenta de forma considerable y, se puede llegar a dar el caso en el que, si la velocidad del objeto es alta puede perderse este de vista.

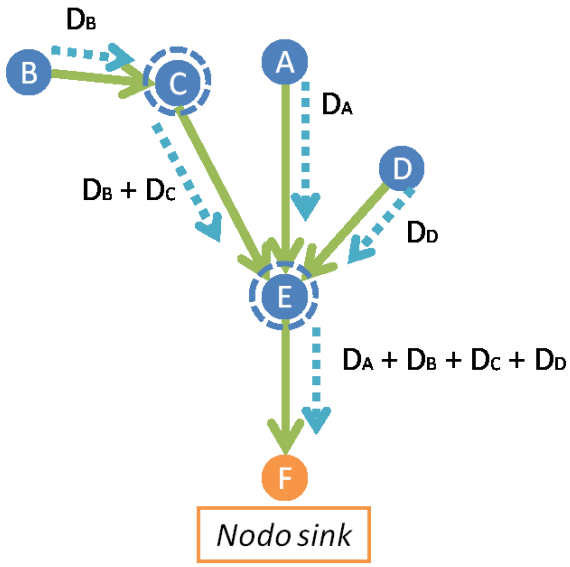


Figura 2.8: Ejemplo de agregación de datos

2.5.2. Fusión de datos

Aunque con los métodos de agregación de datos se puede conseguir reducir el tráfico generado en aplicaciones de detección y seguimiento de blancos, la fusión de datos se establece como una respuesta más eficaz para procesar la información obtenida en el sensado precisamente porque aprovecha la capacidad de procesado de los nodos.

En el caso de un sistema que aplique técnicas de fusión de datos el comportamiento de los nodos en la red será similar a aquel que utilice agregación de datos. Los datos obtenidos serán enviados hacia un nodo destino, que tendrá que procesarlos. Suponiendo una configuración multisalto, cada uno de los paquetes viajarán salto a salto, de un nodo a otro a través de la red. De forma similar a la técnica anterior si un nodo recibe más de un paquete en un mismo instante agrupará en un único mensaje la información contenida en ambos, enviándolo hacia el destino y reduciendo el tráfico presente en la red. Sin embargo, en lugar de simplemente agregar los datos recibidos, procesará tanto las medidas obtenidas por él mismo como las recibidas de sus vecinos

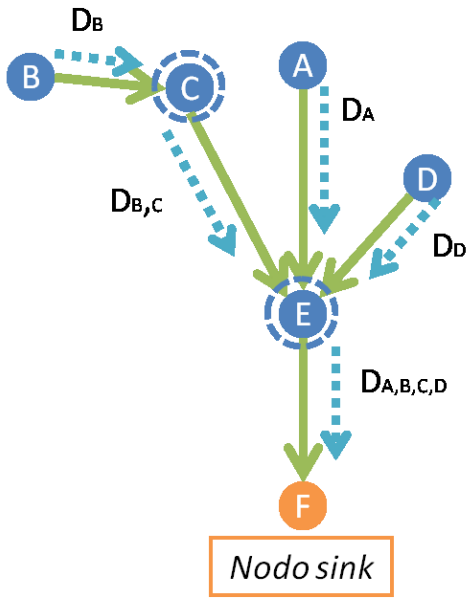


Figura 2.9: Ejemplo de fusión de datos

combinando la información y los datos de forma que genere un único paquete.

Utilizando la redundancia de información de los datos obtenidos, las técnicas de fusión de información pueden reducir aún más el tráfico presente en el sistema para un mismo esquema de red tal y como puede observarse en la Figura 2.9. El tamaño de los paquetes enviados se reduce drásticamente comparado con aquellos procedentes de la agregación de datos. En este caso, suponiendo que un nodo recibe más de un paquete de idéntico tamaño en el mismo instante, extrae la información útil de ambos, procesándola y fusionándola, generando a su salida un paquete de el mismo peso que cualquiera de los recibidos - o drásticamente menor en caso de que realice una estimación de la posición - que enviará hacia el nodo destino.

Sin embargo, es necesario tener en cuenta que la forma de combinar los datos va a depender del tipo de información transmitida, de lo que se pretenda determinar en el nodo destino y de lo que el nodo *sink* pueda necesitar, pues esta información puede no ser ahora la única necesaria para que se realice el procesado final de los datos. De lo contrario pueden aparecer una serie de problemas que impliquen cálculos erróneos en destino.

Como ejemplo, una red de sensores con esquema centralizado. Todos los nodos de la red se

despiertan periódicamente al mismo tiempo para escuchar enviando esta información a través de la red hasta el nodo destino. Este nodo realiza el procesado final de la información para obtener una estimación. Utilizar agregación de datos no supone ningún problema a priori, únicamente puede estar en la cantidad de información redundante que el nodo destino recibe y debe procesar. La fusión de datos tal y como se ha descrito aquí reporta, sin embargo, un gran problema, pues la dificultad de saber en cada uno de los saltos de dónde proviene esa estimación, qué nodos estaban implicados y cuándo se ha realizado nos pueden llevar estimaciones erróneas de la posición del blanco.

Se hace necesario realizar un estudio sobre posibles algoritmos que sean aplicables y reporten una solución viable para posibilitar el desarrollo de sistemas que apliquen técnicas de fusión de datos y en los que el nodo destino realice un procesado final de estos para obtener la estimación de la posición y velocidad del blanco en cada instante. En posteriores capítulos se describen algunos de estos posibles algoritmos estableciendo cuál podría ser su aplicación en este área concreta.

Capítulo 3

MODELO DE SISTEMA

En este capítulo se detalla el modelo de sistema utilizado a lo largo de este proyecto. Abarca tanto el modelo de movimiento que describe el blanco como el modelo de sensor presente en cada uno de los nodos de la red. Asimismo se introducen las distintas arquitecturas del sistema estudiadas y el modelo de red que finalmente ha sido objeto de estudio.

3.1. Modelo dinámico

La descripción del modelo de movimiento es vital para las aplicaciones de seguimiento [8]. Las principales señales de interés en este tipo de aplicaciones están relacionadas con la posición $\mathbf{p}_k$, la velocidad $\mathbf{v}_k$ y la aceleración $\mathbf{a}_k$ en un instante k.

De forma general, se puede describir el estado del blanco a través de un sistema dinámico en el que se asume que la velocidad $\mathbf{v}_k$ está sujeta a una aceleración desconocida, de forma que en el instante k :

$$\mathbf{x}_k = f\left(\mathbf{x}_{k-1},\, \mathbf{u}_k\right) \tag{3.1}$$

$$\mathbf{z}_k = g\left(\mathbf{x}_k,\, \mathbf{w}_k\right) \tag{3.2}$$

donde, como se puede ver en (3.1), el estado del blanco $\mathbf{x}_k$ es función del estado en el instante anterior $\mathbf{x}_{k-1}$ y del ruido en ese instante $\mathbf{u}_k$. La medida del sensor en ese instante $\mathbf{z}_k$ es estadísticamente dependiente del estado del blanco que ha de ser estimado, es decir, es función del estado del blanco en ese instante $\mathbf{x}_k$ más una componente caracterizada por el ruido de observación $\mathbf{w}_k$.

Se asume que ambos ruidos, $\mathbf{u}_k$ y $\mathbf{w}_k$, son estadísticamente independientes.

Al ser una aplicación de localización y seguimiento, el vector $\mathbf{x}_k$ estará caracterizado por cuatro elementos reales (3.3): dos para establecer la posición dentro de un sistema de coordenadas cartesianas y otros dos para determinar su velocidad.

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_x \\ \mathbf{v}_x \end{bmatrix} \rightarrow \mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} \tag{3.3}$$

Utilizando las ecuaciones diferenciales $\dot{\mathbf{p}}_k = \mathbf{v}_k$ y $\dot{\mathbf{v}}_k = \mathbf{a}_k$ se pueden obtener distintas relaciones como $\mathbf{p}_k = \mathbf{p}_0 + \mathbf{v}_0 \cdot k$ (si se considera una velocidad constante) o $\mathbf{p}_k = \mathbf{p}_0 + \mathbf{v}_0 k + \mathbf{a}_0 \cdot k^2$ (si es en este caso la aceleración la que es constante). Si además consideramos un periodo de muestreo T_s, obtenemos un modelo de movimiento discreto en el tiempo entre dos medidas obtenidas en instantes consecutivos [8].

Si se asume que la velocidad $\mathbf{v}_k$ es medible, la posición del blanco en un instante k puede modelarse como:

$$\mathbf{p}_k = \mathbf{p}_{k-1} + T_s \cdot \mathbf{v}_k + \frac{T_s^2}{2} \cdot \mathbf{u}_k, \tag{3.4}$$

donde $\mathbf{u}_k$ es función del ruido en ese instante.

Lo que, considerando tanto 3.4 como 3.1 puede describirse como:

$$\underbrace{\begin{pmatrix} \mathbf{p}_k \\ \mathbf{v}_k \end{pmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{pmatrix} I & T_s \cdot I \\ 0 & I \end{pmatrix}}_{\mathbf{G}_x} \underbrace{\begin{pmatrix} \mathbf{p}_{k-1} \\ \mathbf{v}_{k-1} \end{pmatrix}}_{\mathbf{x}_{k-1}} + \underbrace{\begin{pmatrix} \frac{T_s^2}{2} \cdot I \\ T_s \cdot I \end{pmatrix}}_{\mathbf{G}_u} \cdot \mathbf{u}_k \tag{3.5}$$

Con todo ello se asume que el modelo es lineal y dinámico en el tiempo [6] y está definido por la expresión (3.6) :

$$\mathbf{x}_k = \mathbf{G}_x \, \mathbf{x}_{k-1} \; + \; \mathbf{G}_u \, \mathbf{u}_k \tag{3.6}$$

donde $\mathbf{G}_x$ y $\mathbf{G}_u$ son matrices definidas por (3.7) y (3.8)

$$\mathbf{G}_x = \begin{pmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.7) \qquad \mathbf{G}_u = \begin{pmatrix} \frac{T_s^2}{2} & 0 \\ 0 & \frac{T_s^2}{2} \\ T_s & 0 \\ 0 & T_s \end{pmatrix} \quad (3.8)$$

siendo T_s el periodo de muestreo del sensor, y $\mathbf{u}_k$ un vector bidimensional que representa un proceso Gaussiano de media nula y matriz de covarianzas $\mathbf{C}_u = \text{diag}\left(\sigma_{u1}^2, \sigma_{u2}^2\right)$.

3.2. Modelo de sensor

El modelo de sensor utilizado es un sensor de potencia acústica. Mide la potencia de la señal de entrada, *RSS(Received Singal Strength)*, estimando la posición del objeto a través de un modelo estadístico que determina la potencia recibida a una cierta distancia. Este modelo asume que:

- El objeto es una fuente puntual.
- Las señales acústicas se propagan sin pérdidas y de igual forma en todas las direcciones (señales isotrópicas).
- Utiliza un modelo de distribución log-normal *(Log-normal Shadowing Model)*.

La principal razón por la que se utiliza este modelo de sensor es que éste no requiere ningún hardware adicional pues la indicación RSS *(RSSI - Received Signal Strength Indication)* es una característica estándar de cualquier sistema de comunicaciones. Esta ventaja supone que ni el coste ni el tamaño del sensor se vean incrementados, además de que, con este método el impacto en el consumo de energía local no resulta significativo [6].

Ya que la señal de potencia recibida puede verse atenuada en mayor o menor medida dependiendo de los efectos introducidos por las características del entorno, y que dicha atenuación puede ser mayor incluso que la causada por la distancia, se ha considerado utilizar un modelo de distribución log-normal para modelar dichas pérdidas [9]. Algunos de estos efectos indeseados están causados por fenómenos de difracción, scattering, reflexiones u obstáculos presentes en el

camino.

Si las pérdidas sufridas por la potencia de la señal a lo largo del camino de comunicación decrecen de forma logarítmica con la distancia *(Log-distance Path Loss Model)*, se pueden expresar como función de esta utilizando un exponente de pérdidas en el camino. Expresado en dB:

$$\bar{PL}(dB) = \bar{PL}\,(d_0) + 10\alpha \log\left(\frac{d}{d_0}\right) \tag{3.9}$$

donde:

- $\bar{PL}\,(d_0)$ es la potencia recibida (dBm) a una distancia de referencia d_0.
- α es el exponente de pérdidas en el camino que define el ratio con el que la potencia decrece con la distancia. Este factor se determina de forma experimental. Su valor dependerá del entorno de propagación en el que el sensor se encuentre desplegado; típicamente para entornos en el interior es de 4 mientras que en el exterior pueden variar entre 2 y 3.
- d distancia a la que se estiman las pérdidas de potencia de la señal.

Sin embargo, este modelo no considera que el entorno de propagación físicamente próximo a un sensor puede diferir en gran medida del entorno próximo de otro sensor. Diferencia que llevaría a la situación en la que en dos sistemas de comunicaciones distintos separados una misma distancia pero cuyos transmisores o receptores estén en distintas localizaciones, existan diferencias significativas en las pérdidas sufridas. En estos casos se utiliza un modelo de distribución normal, añadiendo una variable aleatoria Gaussiana, X_σ (en dB), de media nula y desviación estándar σ [9].

$$\bar{PL}(dB) = \bar{PL}\,(d_0) + 10\alpha \log\left(\frac{d}{d_0}\right) + X_\sigma \tag{3.10}$$

donde en la práctica, tanto los valores de α como de σ son determinados de forma experimental.

Así, considerando N sensores desplegados en la red, la potencia recibida por el sensor n en el instante k será función de la posición $\mathbf{p}_k = [x_k, y_k]^T$, como se puede observar en la expresión (3.11), [6].

$$z_k^n = P_0 - 10\alpha \log\left(\frac{\|\mathbf{p}_k - \mathbf{r}_n\|}{d_0}\right) + w_k^n \tag{3.11}$$

donde:

- P_0 es la potencia recibida (dBm) a una distancia de referencia d_0, que se relaciona con la potencia transmitida como $P_0 = P_{tx} - \bar{PL}\,(d_0)$.
- w_k^n es la potencia del ruido de observación del nodo n, modelada como una variable aleatoria Gaussiana (dB) de media nula y varianza σ_w^2.
- $\mathbf{r}_n$ es la posición del sensor n.
- $\|\cdot\|$ es la distancia euclídea.

3.3. Arquitectura del sistema

Uno de los objetivos principales a tener en cuenta en el diseño de una red de sensores inalámbricos es la maximización de su vida útil reduciendo todo lo posible tanto el gasto computacional como la comunicación entre nodos. Ya que es esta última característica de la red la que produce mayor gasto de energía, el diseño de la arquitectura del sistema influye en gran medida en la eficiencia energética del seguimiento del objeto.

Dentro de los escenarios típicamente propuestos se pueden distinguir principalmente dos esquemas bien diferenciados: un esquema centralizado y otro descentralizado o de líder. El primero se caracteriza por la presencia de un único nodo central o nodo destino que, además de recibir todas las medidas obtenidas por cada uno de los nodos de la red, será el encargado de realizar tanto las tareas de computación como las de estimación de la posición del blanco en cada instante de tiempo. El esquema descentralizado sin embargo, no posee un único nodo central, sino que, o bien varios nodos de la red, o bien todos los nodos tendrán capacidades de computación además de las de sensado y estimación, enviando la información procesada al nodo destino.

A continuación se detalla cuál es el funcionamiento de la red en cada caso, así como el modelo de arquitectura del sistema elegido en el desarrollo de este estudio.

3.3.1. Esquema Centralizado

Un esquema centralizado es el diseño más básico de una red de sensores. Consideramos una red formada por N sensores que se encuentran distribuidos en un área geográfica determinada, donde cada uno toma una observación o medida que depende tanto de su localización como de

la posición del blanco dentro de la zona de cobertura de la red. Existe en esta un único nodo destino o *nodo sink* que se encarga de recibir la información y estimar la posición del blanco a través de su capacidad de cálculo.

Suponemos además que en este sistema todos los nodos son despertados al mismo tiempo, cada periodo de tiempo, T_s *(Periodo de muestreo o sampling)*. Estos, una vez han realizado las tareas de sensado, envían la infomación recibida al nodo central sin realizar ningún tipo de procesado sobre la observación obtenida. El nodo destino obtendrá por tanto N medidas distintas cada vez, que habrá de procesar. Considerando que cada nodo existente en la red conoce la localización de sus vecinos y del *sink*, y que envía la información a aquel vecino más cernano al nodo destino, en el mejor de los casos se enviarán $N-1$ paquetes cada instante de muestreo, T_s, de igual tamaño que circularán a través de la red. Un ejemplo de este tipo de arquitecturas del sistema se muestra en la Figura 3.1, donde la red de sensores sigue una topología en rejilla.

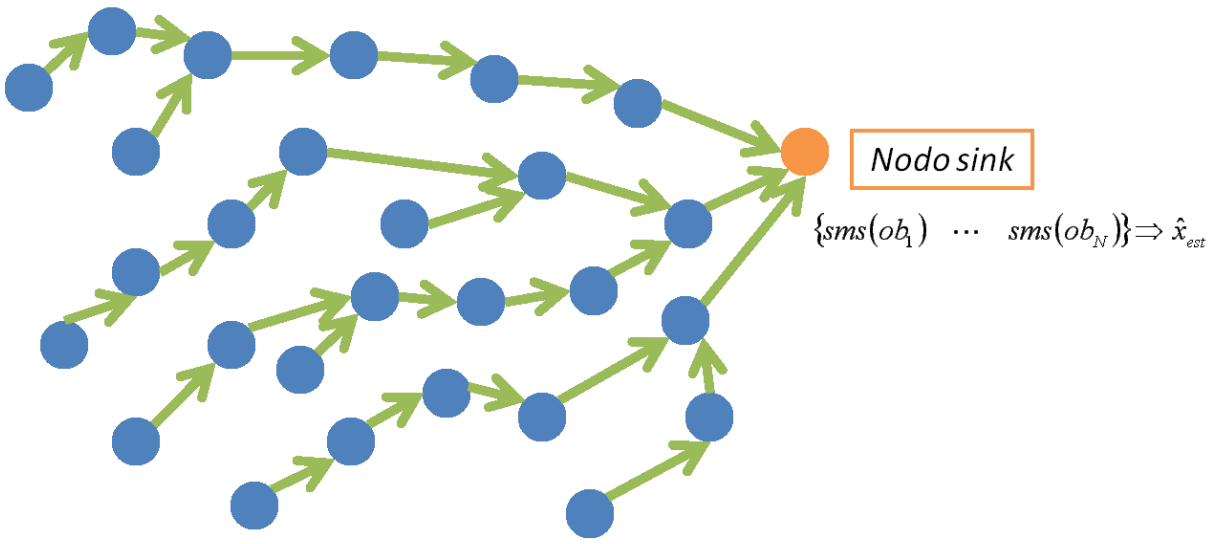


Figura 3.1: Ejemplo del paso de mensajes en un sistema centralizado

Aunque este esquema resulta muy sencillo de implementar, el principal problema de los sistemas con un esquema centralizado es la gran capacidad de computación necesaria en el nodo destino. Esta aumentará considerablemente con el número de nodos existentes de la red, por lo que su escalabilidad es baja. Además, la gran cantidad de mensajes que son enviados a través de la red hace que la probabilidad de colisión e interferencia en el medio aumente considerablemente.

Así, su consumo alto de energía limita en gran medida la vida útil del sistema.

3.3.2. Esquema Descentralizado o de Líder

En un esquema descentralizado, el procesado de la información obtenida en cada instante de observación no se realiza únicamente en el nodo destino. En un sistema básico de este tipo se seleccionan inicialmente algunos nodos, estratégicamente situados dentro del área que abarca la red y que se despiertan periódicamente, cada T_s, y escuchan. Si la medida realizada sobrepasa un determinado umbral de potencia en alguno de ellos entonces se decide que un objeto ha sido detectado. Dicho nodo se establece inicialmente como *líder* de la red para ese instante k y comienza el algoritmo de enrutado de información.

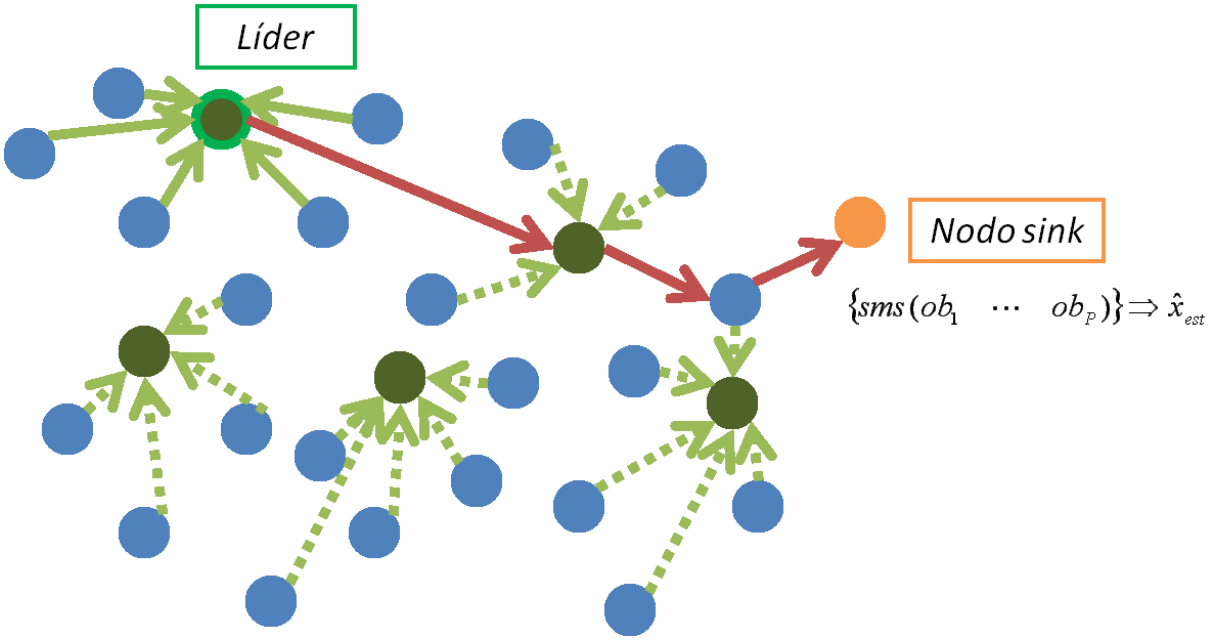


Figura 3.2: Ejemplo del paso de mensajes en un sistema de líder

El nodo *líder* despierta a sus $P-1$ nodos vecinos que le envían sus medidas sin realizar ningún tipo de procesado siendo el primero el que las agrupa en paquetes y las envía al nodo destino. El nodo destino (*nodo sink*) se encarga, al igual que ocurría en el esquema centralizado anterior, de procesar los datos y estimar la posición del objeto. La diferencia está en que ahora sólo ha de procesar P medidas ($P << N$), lo que se traduce en un ahorro considerable de recursos y de tiempo de procesado. Un ejemplo de funcionamiento de este tipo de arquitectura del

sistema se puede observar en la Figura 3.2 donde las flechas color verde indican intercambio de información de las observaciones obtenidas, y las flechas rojas el enrutado del paquete obtenido en la agregación de estas.

La ventaja principal de esta configuración radica en que solo una parte de la red permanece activa, optimizando la energía disponible en el sistema. La figura del *líder* se modifica con el movimiento del blanco, siendo el nuevo *líder* aquel nodo activo (esto es, vecino o nodo líder actual) que haya obtenido un mayor nivel de potencia en su medida y estime tener suficiente energía.

3.3.3. Esquema Distribuido

El esquema inicialmente propuesto para el desarrollo de este proyecto de fin de carrera está en cierto grado basado en los dos anteriormente expuestos. Por un lado, al igual que el esquema centralizado, todos los nodos de la red son despertados al mismo tiempo, cada periodo de muestreo T_s, para realizar las tareas de sensado. Existe también un único nodo encargado de procesar la información enviada por los nodos activos de la red y determinar la estimación de la posición del blanco. Sin embargo, para tratar de evitar el inconveniente principal que este tipo de arquitectura posee, esto es, el número elevado de mensajes enrutados a través de la red hacia el nodo central, se han adoptado ciertas características propias del esquema descentralizado.

Así, la red propuesta se divide inicialmente en *clusters* o grupos de nodos, en cada uno de los cuales se elige un nodo como líder del *cluster*, también llamado *cluster-head*. Éste, cada T_s, despierta a los nodos que conforman su *cluster* que, una vez realizadas las tareas de sensado, le envían las observaciones obtenidas. Cada *cluster-head* realiza el procesado de las observaciones creando un único paquete con la información relevante que mandará a través de la red hacia el nodo destino. Así, el número de paquetes remitidos en cada instante hacia el nodo destino se reduce de N paquetes enviados en el esquema centralizado (entendiendo que la red tiene N nodos desplegados) a C paquetes, siendo C el número de *clusters* en los que se divide la red. El funcionamiento de este tipo de arquitectura de red, se puede observar en la Figura 3.3, con la misma topología de rejilla existente en los ejemplos anteriores.

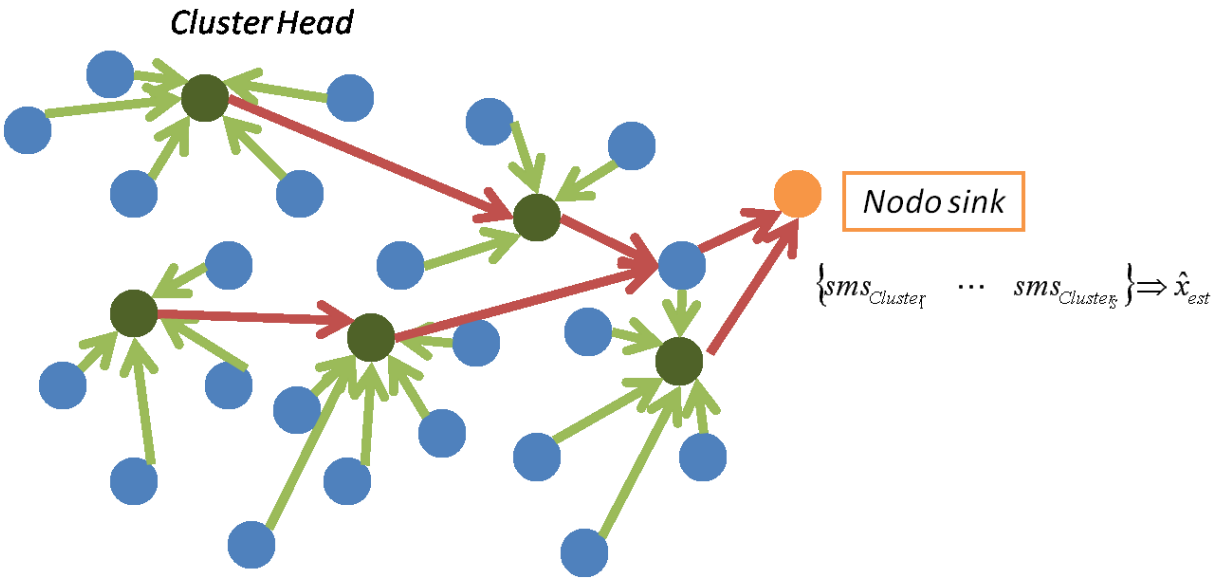


Figura 3.3: Ejemplo del paso de mensajes en un sistema de líder

Además, para reducir aún más el número de paquetes, esta arquitectura permite la utilización de técnicas de agregación o de fusión de datos. En caso de aplicar la primera se reducirá el número de paquetes circulando por la red pero el tamaño de estos se verá aumentado. Si consideramos la fusión de datos lograremos reducir el número de paquetes que viajan por la red manteniendo el tamaño de los mismos. Realizando fusión de datos cuando uno de los nodos de la red recibe un paquete y en su cola de espera se encuentra otro creado en el mismo instante de tiempo procesa la información que contienen ambos mensajes, fusionándola y creando un único paquete, que sustituye por aquel que esperaba a ser enviado.

Existen otras técnicas que pueden ser aplicadas para intentar alargar la vida útil de una red de sensores, como la forma en la que se enrutan los paquetes o la reconfiguración de la red en caso de que uno de los nodos de esta se quede sin batería muera para el resto de los nodos. Las decisiones que sobre estos aspectos se tomen en las simulaciones realizadas para este estudio, se describen en los *Capítulos 5 y 6.*

Capítulo 4

SEGUIMIENTO DE BLANCOS

En este capítulo se introduce la formulación necesaria para desarrollar la solución al problema de estimación de la posición y velocidad del blanco. En la Sección 4.1 se desarrolla la aproximación Bayesiana utilizada en su resolución; en la Sección 4.2 las nociones básicas de los filtros de partículas, para en 4.4 mostrar una descripción de los filtros de partículas distribuidos, objeto de estudio en este Proyecto de Fin de Carrera.

4.1. Estimación Bayesiana

Para resolver el problema de localización y seguimiento se pretende estimar la posición del objetivo, $\mathbf{x}_k$ en un instante k a partir del histórico de una serie de medidas obtenidas por el sensor $\bar{z}_k$. En el caso de este estudio se utilizará la aproximación Bayesiana en su resolución [10].

La notación estadística que utilizaremos a lo largo de este estudio toma:

- $p(\mathbf{x})$ como la función de distribución de probabilidad a priori, FDP, sobre el estado $\mathbf{x}$.
- $p(z|\mathbf{x})$ como la función de probabilidad de z dada $\mathbf{x}$, también llamada *función de verosimilitud.*
- $p(\mathbf{x}|z)$ como la función de probabilidad a posteriori de $\mathbf{x}$ dada una medida z o *convicción.*
- $p(z)$ como la distribución marginal.

Utilizando el Teorema de Bayes se puede relacionar la distribución a priori $p(\mathbf{x})$, la verosimilitud $p(z|\mathbf{x})$ y la función de probabilidad a posteriori $p(\mathbf{x}|z)$ como:

$$p(\mathbf{x}|z) = \frac{p(z|\mathbf{x})p(\mathbf{x})}{\int p(z|\mathbf{x})p(\mathbf{x})d\mathbf{x}} = \frac{p(z|\mathbf{x})p(\mathbf{x})}{p(z)} \tag{4.1}$$

La distribución marginal $p(z)$ puede verse como una constante, pudiendo escribir la ecuación anterior (4.1) como:

$$p(\mathbf{x}|z) = \mathbf{C} \cdot p(z|\mathbf{x})p(\mathbf{x}) \tag{4.2}$$

$$p(\mathbf{x}|z) \propto p(z|\mathbf{x})p(\mathbf{x}) \tag{4.3}$$

Para resolver el problema de localización hemos de obtener una aproximación lo más precisa posible de la posición del objeto, $\mathbf{x}_k$, en el instante k. Para ello contamos con el histórico de las medidas realizadas por un sensor desde el instante inicial hasta este instante k, $\bar{\mathbf{z}}_k = \{z_0, z_1, \cdots, z_k\}$, donde la medida realizada en el instante k es z_k. Por tanto se ha de determinar $\hat{\mathbf{x}}_k\left(\bar{\mathbf{z}}_k\right)$, para lo que se minimizará el coste medio:

$$\epsilon = E\left[d\left(\hat{\mathbf{x}}_k\left(\bar{\mathbf{z}}_k\right), \mathbf{x}_k\right)\right] \tag{4.4}$$

donde $d\left(\cdot,\cdot\right)$ es una función de pérdidas que mide el funcionamiento del estimador.

Trabajaremos con el estimador de mínimo error cuadrático medio *(MMSE - Mimimum Mean Square Error)*, ecuación (4.5), para lo cual deberemos obtener la distribución a posteriori $p\left(\mathbf{x}_k|\hat{\mathbf{x}}_k\right)$ de una forma eficiente.

$$\hat{\mathbf{x}}_k = E\left[\mathbf{x}_k|\bar{\mathbf{z}}_k\right] = \int \mathbf{x}_k \cdot p\left(\mathbf{x}_k|\bar{\mathbf{z}}_k\right) \cdot d\mathbf{x}_k \tag{4.5}$$

Ya que la medida z se obtiene a lo largo del tiempo, $\bar{\mathbf{z}}_k$, podemos realizar una ampliación Bayesiana de la ecuación (4.3) para incluir el histórico de las medidas realizadas. Si en el instante k un sensor posee una *convicción* previa dada por el histórico de medidas anteriores $p\left(\mathbf{x}_k|\bar{\mathbf{z}}_{k-1}\right)$, y en ese instante se realiza una nueva medida, z_k, entonces:

$$p\left(\mathbf{x}_k|\bar{\mathbf{z}}_k\right) \propto p\left(z_k|\mathbf{x}_k\right) p\left(\mathbf{x}_k|\bar{\mathbf{z}}_{k-1}\right) \tag{4.6}$$

$$p\left(\mathbf{x}_k|\bar{\mathbf{z}}_k\right) = p\left(z_k|\mathbf{x}_k\right) \int p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) p\left(\mathbf{x}_{k-1}|\bar{\mathbf{z}}_{k-1}\right) d\mathbf{x}_{k-1} \tag{4.7}$$

donde:

- $p\left(z_k|\mathbf{x}_k\right)$ es la probabilidad de observación dada por la posición del objeto.
- $p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right)$ está relacionado con la dinámica del vehículo.
- $p\left(\mathbf{x}_{k-1}|\bar{\mathbf{z}}_{k-1}\right)$ es la distribución de probabilidad a posteriori heredada del paso anterior.

4.2. Filtro de partículas

Para lograr una resolución del problema de detección y seguimiento de un objeto en la red de sensores se pretende realizar un procesado secuencial de la señal obtenida en el sensor, es decir, se desea una estimación recursiva del estado del blanco, $\mathbf{x}_k$, utilizando para ello las observaciones $\bar{\mathbf{z}}_k$ tomadas por el sensor. La implementación del algoritmo desarrollado por la ecuación (4.7) resulta imposible en la mayor parte de las aplicaciones prácticas pues la actualización de las distribuciones pueden requerir integraciones que no pueden llevarse a cabo de forma analítica o cuyo coste computacional resulta demasiado elevado. [11].

Desarrollando la ecuación (4.6) de forma recursiva, a través de la siguiente relación:

$$p\left(\mathbf{x}_k|\bar{\mathbf{z}}_k\right) = \frac{p\left(z_k|\mathbf{x}_k\right)p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right)}{p\left(z_k|\bar{\mathbf{z}}_{k-1}\right)}p\left(\mathbf{x}_{k-1}|\bar{\mathbf{z}}_{k-1}\right), \tag{4.8}$$

obtenemos que:

$$p\left(\mathbf{x}_k|\bar{\mathbf{z}}_k\right) \propto p\left(\mathbf{x}_0|z_0\right)\prod_{i=0}^{k} p\left(z_i|\mathbf{x}_i\right)p\left(\mathbf{x}_i|\mathbf{x}_{i-1}\right) \tag{4.9}$$

Ya que pasar de $p\left(\bar{\mathbf{x}}_{k-1}|\bar{\mathbf{z}}_{k-1}\right)$ a $p\left(\bar{\mathbf{x}}_k|\bar{\mathbf{z}}_k\right)$ suele ser extremadamente complejo de obtener de forma analítica, usaremos métodos de cálculo basados en aproximaciones.

El filtrado de partículas se engloba dentro de las metodologías secuenciales de Monte Carlo. Su idea básica es la computación recursiva de distribuciones de probabilidad utilizando conceptos como la importancia del muestreo y realizando aproximación de distribuciones de probabilidad con medidas aleatorias discretas. Se trata de una aproximación numérica que plantea una solución efectiva a diversos problemas estadísticos, como en la detección y seguimiento de blancos en movimiento.

En un filtro de partículas las distribuciones continuas son aproximadas como medidas aleatorias discretas compuestas por partículas asociadas a pesos. Cada partícula es una muestra de un

estado desconocido dentro del espacio de estados. Los pesos asociados a ellas forman *masas o grupos de probabilidad* calculados utilizando la teoría Bayesiana. Como resultado, una distribución $p(x_k)$, puede ser aproximada como:

$$\chi = \left\{ x_k^{(m)}, w_k^{(m)} \right\}_{m=1}^{M} \tag{4.10}$$

donde $x_k^{(m)}$ son las partículas y $w_k^{(m)}$ sus pesos asociados, siendo M el número de partículas utilizadas en la aproximación. En este caso χ aproxima la distribución $p(x_k)$ a través de la siguiente ecuación:

$$p(x_k) \approx \sum_{m=1}^{M} w_k^{(m)} \cdot \delta \left(x_k - x_k^{(m)} \right) \tag{4.11}$$

donde $\delta(\cdot)$ es la función de Dirac. A través de esta aproximación el cálculo de las estimaciones, que conllevaba integraciones complejas, se puede simplificar usando sumatorios. Así, por ejemplo:

$$E\left(g\left(X\right)\right) = \int g(x)p(x)dx \tag{4.12}$$

puede aproximarse como:

$$E\left(g\left(X\right)\right) \approx \sum_{m=1}^{M} w^{(m)} g(x^{(m)}). \tag{4.13}$$

En la Figura 4.1 podemos observar un ejemplo de la aproximación de una distribución de probabilidad continua a través de un filtro de partículas. En la posición inicial, 1, las partículas - *círculos amarillos* -, se distribuyen teniendo en cuenta la función de distribución inicial del problema. En 2 se pueden observar estas tras la asignación y normalización de sus pesos asociados, que equivaldrían al diámetro de las circunferencias - *círculos rojos* -.

Otro concepto a introducir dentro de estas metodologías es la importancia del muestreo *(importance sampling)*. Suponiendo que queremos aproximar la distribución, $p(x)$, podemos generar las partículas, inicializándolas con idénticos pesos, $1/M$. Cuando el muestreo directo de $p(x)$ es impracticable, se pueden generar las partículas $x^{(m)}$ utilizando una distribución $\pi(x)$, también conocida como **función de importancia**, asignando pesos de acuerdo a:

$$w^{*(m)} = \frac{p(x)}{\pi(x)} \tag{4.14}$$

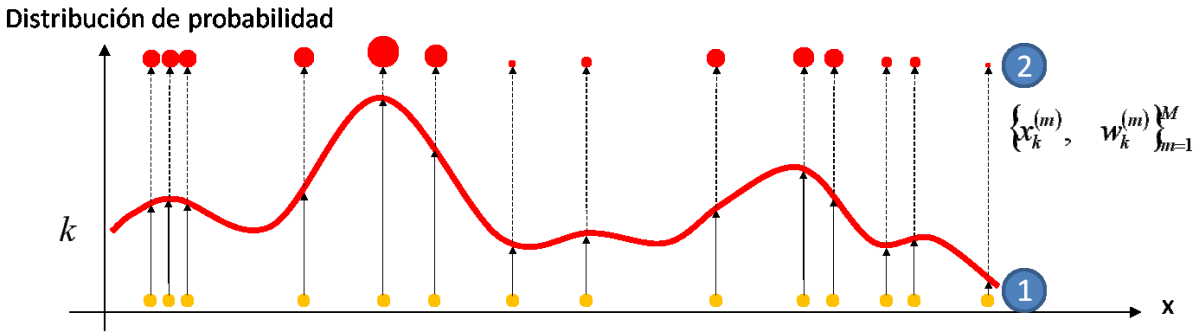


Figura 4.1: Aproximación de una distribución de probabilidad a través de un filtro de partículas: partículas y pesos asociados [4]

que, una vez normalizados se pueden denotar como:

$$w^{(m)} = \frac{w^{*(m)}}{\sum_{i=1}^{M} w^{*(i)}} \tag{4.15}$$

Si aproximamos la distribución a posteriori $p\left(\bar{\mathbf{x}}_{k-1}|\bar{\mathbf{z}}_{k-1}\right)$ por el grupo de partículas $\chi_{k-1} = \left\{\bar{x}_{k-1}^{(m)}, w_{k-1}^{(m)}\right\}_{m=1}^{M}$ la resolución del problema de seguimiento del blanco pasará por calcular χ_k a partir de χ_{k-1} usando la medida obtenida por el sensor en el siguiente instante, z_k.

Si usamos una función de importancia que pueda ser factorizada como:

$$\pi\left(\bar{\mathbf{x}}_k|\bar{\mathbf{z}}_k\right) = \pi\left(\mathbf{x}_k|\bar{\mathbf{x}}_{k-1}, \bar{\mathbf{z}}_k\right)\pi\left(\bar{\mathbf{x}}_{k-1}|\bar{\mathbf{z}}_{k-1}\right) \tag{4.16}$$

y si se cumple que:

$$\bar{\mathbf{x}}_{k-1}^{(m)} \approx \pi\left(\bar{\mathbf{x}}_{k-1}|\bar{\mathbf{z}}_{k-1}\right) \tag{4.17}$$

$$\bar{w}_{k-1}^{(m)} \propto \frac{p\left(\bar{\mathbf{x}}_{k-1}^{(m)}|\bar{\mathbf{z}}_{k-1}\right)}{\pi\left(\bar{\mathbf{x}}_{k-1}^{(m)}|\bar{\mathbf{z}}_{k-1}\right)} \tag{4.18}$$

entonces se puede aumentar la trayectoria $\bar{\mathbf{x}}_{k-1}^{(m)}$ con $\mathbf{x}_k^{(m)} \sim \pi\left(\mathbf{x}_k|\bar{\mathbf{x}}_{k-1}^{(m)}, \bar{\mathbf{z}}_k\right)$ actualizando los pesos asociados de acuerdo a:

$$w_k^{(m)} \propto \frac{p\left(z_k|\mathbf{x}_k^{(m)}\right)p\left(\mathbf{x}_k^{(m)}|\mathbf{x}_{k-1}^{(m)}\right)}{\pi\left(\mathbf{x}_k^{(m)}|\bar{\mathbf{x}}_{k-1}^{(m)}, \bar{\mathbf{z}}_k\right)} w_{k-1}^{(m)} \tag{4.19}$$

En el caso óptimo la función de importancia $\pi\left(\mathbf{x}_k^{(m)}|\bar{\mathbf{x}}_{k-1}^{(m)}, \bar{\mathbf{z}}_k\right)$ se asemeja a la distribución de probabilidad a la que está siendo aproximada $p\left(\mathbf{x}_k^{(m)}|\mathbf{x}_{k-1}^{(m)}\right)$. Esto implica que las actualizaciones de los pesos se puedan llevar a cabo como:

$$w_k^{(m)} \propto p\left(z_k|\mathbf{x}_k^{(m)}\right) w_{k-1}^{(m)} \tag{4.20}$$

4.2.1. Remuestreo

Un gran problema que se ha de tener en cuenta dentro del filtrado de partículas es que, tras la ejecución de varios ciclos del algoritmo, la medición aleatoria discreta degenera rápidamente produciéndose una dispersión de las partículas causada por el progresivo incremento de la varianza de su distribución. Debido a esta dispersión la mayor parte de las partículas se alejan demasiado del valor esperado, asignándoles pesos insignificantes $w_k^{(m)} \to 0$ frente a un reducido grupo con pesos notorios. Todo esto se traduce en un deterioro de la eficiencia del filtro de partículas, como muestra la Figura 4.2.

Para reducir en gran medida esta degeneración se suele realizar remuestreo *(resampling)*. Este fenómeno se puede observar en la Figura 4.3. El remuestreo consiste en la eliminación de aquellas partículas con pequeños pesos y la réplica de aquellas otras con pesos significativos hasta obtener el mismo número de partículas M. Cuanto mayor peso tenga la partícula, más réplicas de esta se obtendrán tras el remuestreo. Una vez finalizado este proceso todos los pesos de las nuevas partículas serán igual a $1/M$.

Los pasos para conseguir este remuestreo se pueden resumir en:

1. Eliminar aquellas partículas con pesos cercanos a 0, o que estén por debajo de un umbral determinado N_{eff}, pasando de tener M partículas a P partículas.
2. Obtener M partículas $x_k^{(m)}$ a través de la réplica de las P partículas restantes según sus pesos asignados.
3. Determinar $x_k^{(m)} = x_k^{*(m)}$ asignando los nuevos pesos $w_k^{*(m)} = 1/M$

Un indicador del grado de degeneración del filtro de partículas es el llamado número de partículas eficaces, N_{eff}, que se define en términos del número de partículas y la varianza de los pesos de estas según la ecuación (5.26), [4].

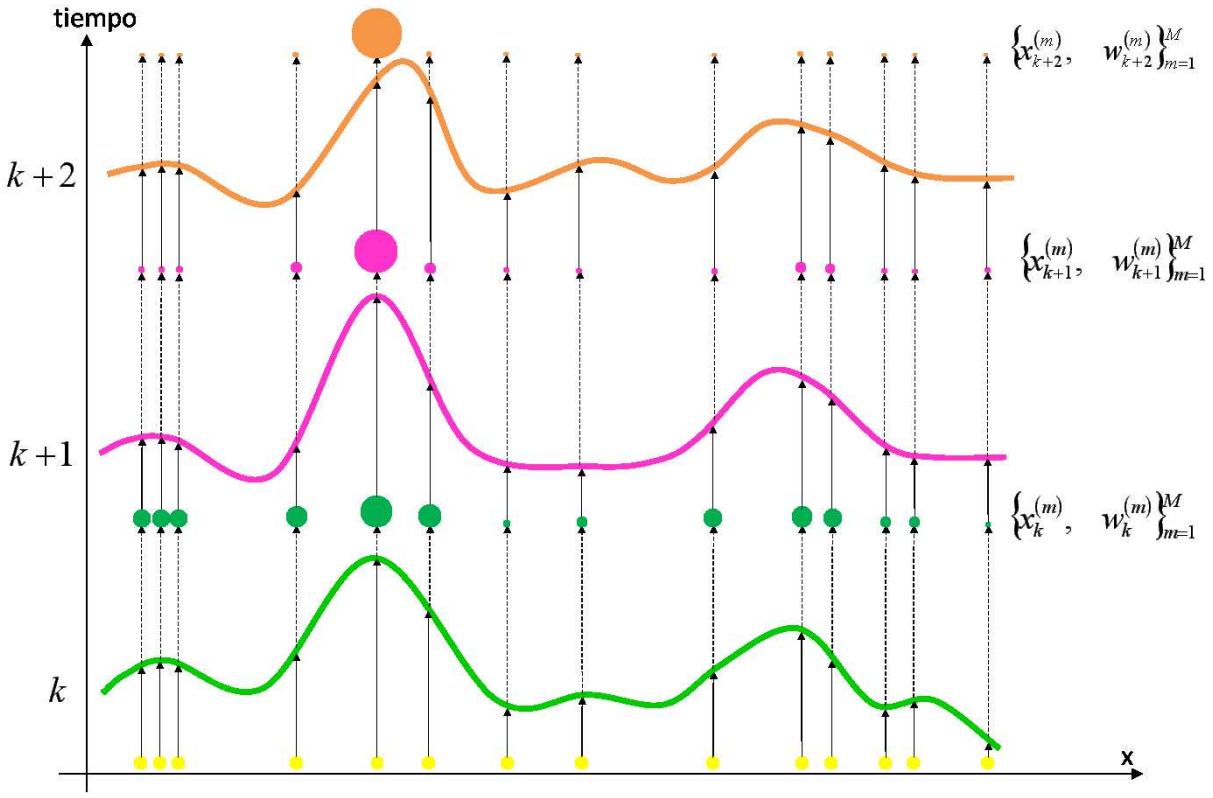


Figura 4.2: Problema en un filtro de partículas: la medida discreta con soporte aleatorio se degenera en unas pocas iteraciones [4]

$$N_{eff} = \frac{M}{1 + M^2 \cdot Var\left(w_k^{(m)}\right)^2} \tag{4.21}$$

donde:

- M es el número de partículas del filtro
- $w_k^{(m)}$ el peso asociados en el instante k a la partícula m.
- Var es la varianza de los pesos.
- se cumple siempre la propiedad $1 < N_{eff} < M$.

El número efectivo de partículas tendrá su máximo en el caso en el que todas las partículas tengan pesos idénticos, $w_k^{(m)} = 1/M$, siendo $N_{eff_{max}} = M$. El límite mínimo lo alcanzará en el su-

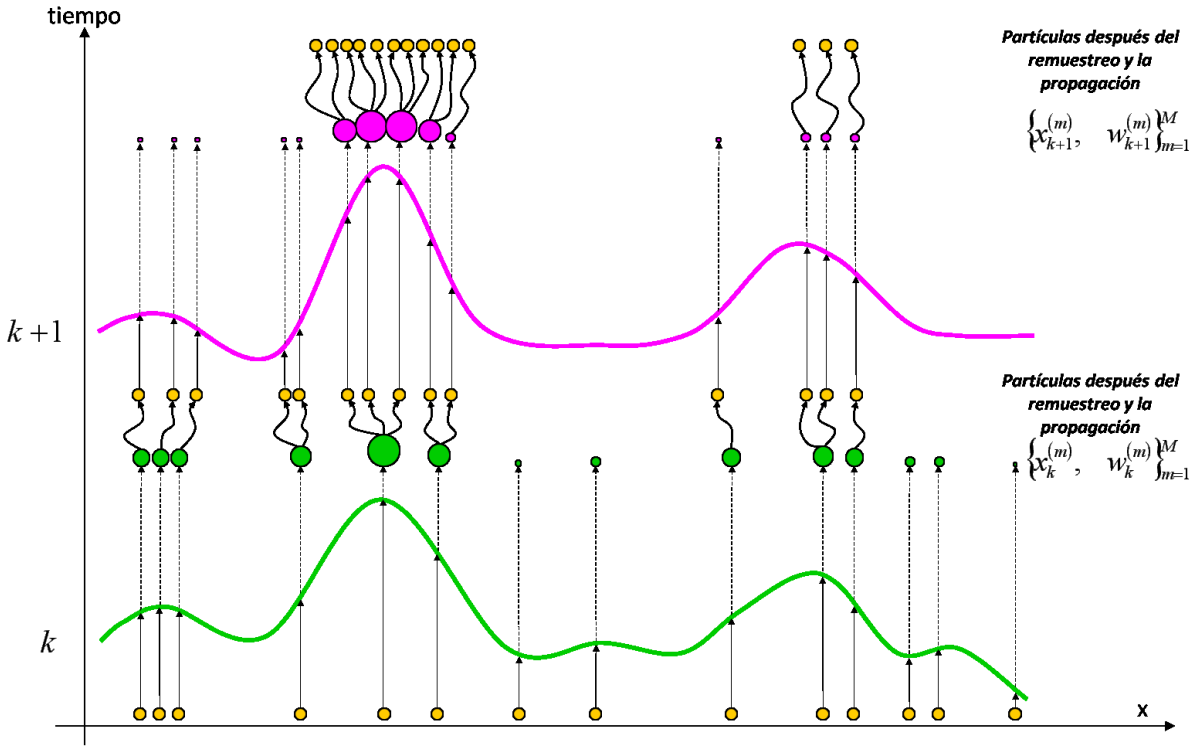


Figura 4.3: Descripción de un filtro de partículas con remuestreo [4]

puesto en el que toda la masa de probabilidad se dirijan hacia una sola partícula, con $N_{eff_{min}} = 1$.

Para simplificar su cálculo, el número se puede determinar de forma aproximada a través de la ecuación (4.22):

$$\hat{N}_{eff} = \frac{1}{\sum_i \left(w_k^{(m)}\right)^2} \tag{4.22}$$

El remuestreo se considera necesario en el caso de que el N_{eff} no alcance un determinado umbral $\left(\hat{N}_{eff} < N_{th}\right)$, que generalmente se define como $N_{th} = 2M/3$.

4.3. Esquema de un filtro de partículas

El funcionamiento del algoritmo de filtrado de partículas descrito en este capítulo puede resumirse como la aproximación de la función de convicción $p\left(\mathbf{x}_k|\overline{\mathbf{z}_k}\right)$, que en un problema de localización y seguimiento de un blanco se corresponde a la posición y velocidad del objeto, a

través de un conjunto de partículas y sus pesos [6],[11]:

$$\begin{aligned} \mathbf{x}_k^{(m)}: \quad & \mathbf{x}_k^{(m)} = \left[x_{1,k}^{(m)},\, x_{2,k}^{(m)},\, \dot{x}_{1,k}^{(m)},\, \dot{x}_{2,k}^{(m)}\right] \quad m = 1, 2, \ldots, M \\ w_k^{(m)}: \quad & normalizados \quad \sum_{m=1}^{M} w_k^{(m)} = 1 \\ & \Rightarrow p\left(\mathbf{x}_k|\overline{\mathbf{z}_k}\right) = \sum_{m=1}^{M} w_k^{(m)} \cdot \delta\left(\mathbf{x}_k - \mathbf{x}_k^{(m)}\right) \end{aligned} \tag{4.23}$$

La secuencia de pasos de dicho algoritmo se resume como:

1. ***Inicialización:*** Las M partículas, $\mathbf{x}_0^{(m)}$, son inicializadas a través de una función de distribución $\pi\left(\mathbf{x}_0^{(m)}\right)$ (distribución uniforme en el área de la red de sensores) y sus correspondientes pesos a $w_0^{(m)} = 1/M$

2. ***Generación de nuevas partículas:*** Cuando llegan nuevas observaciones se aumenta el conjunto de observaciones, esto es, se generan nuevas partículas de acuerdo con la función de distribución $p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right)$:

$$\mathbf{x}_{k-1}^{(m)} \rightarrow \mathbf{x}_k^{(m)} \quad \mathbf{x}_k^{(m)} = f\left(\mathbf{x}_{k-1}^{(m)}\right) + \mathbf{n}_k \tag{4.24}$$

   donde $\mathbf{n}_k$ es el componente de ruido, de distribución Normal, media nula y varianza σ_u^2.

3. ***Actualización y normalización de los pesos*** Se calculan los nuevos pesos, normalizándolos después para que $\sum_{m=1}^{M} w_k^{(m)} = 1$:

$$w_{k-1}^{(m)} \rightarrow w_k^{(m)} w_k^{(m)} = w_{k-1}^{(m)} p\left(\mathbf{z}_k|\mathbf{x}_k^{(m)}\right) \tag{4.25}$$

4. ***Estimación de la posición:*** Se realiza la estimación de la posición del blanco intentando minimizar el error cuadrático medio (MMSE):

$$\begin{aligned} \widehat{\mathbf{x}}_k &= E[\mathbf{x}_k|\overline{\mathbf{z}_k}] \\ &= \int \mathbf{x}_k p\left(\mathbf{x}_k|\overline{\mathbf{z}_k}\right) d\mathbf{x}_k \end{aligned} \tag{4.26}$$

   aproximando (4.26) como

$$\widehat{\mathbf{x}}_k \approx \sum_{m=1}^{M} w_k^{(m)} \mathbf{x}_k^{(m)} \tag{4.27}$$

5. ***Remuestreo:*** Se remuestrea. Este paso es importante para evitar una degeneración de las medidas aleatorias. Se basa en la réplica de las partículas con mayor peso, eliminando las muestras con menor peso y remuestreando las de mayor peso para obtener de nuevo M observaciones con pesos $w_k^{(m)} = 1/M$.

6. Se continua en el siguiente instante, $k = k + 1$, volviendo al paso 2 para la generación de nuevas partículas.

La actualización de los coeficientes de forma recursiva posibilita que el algoritmo vaya convergiendo con cada iteración, tal y como se muestra en la Figura 4.4.

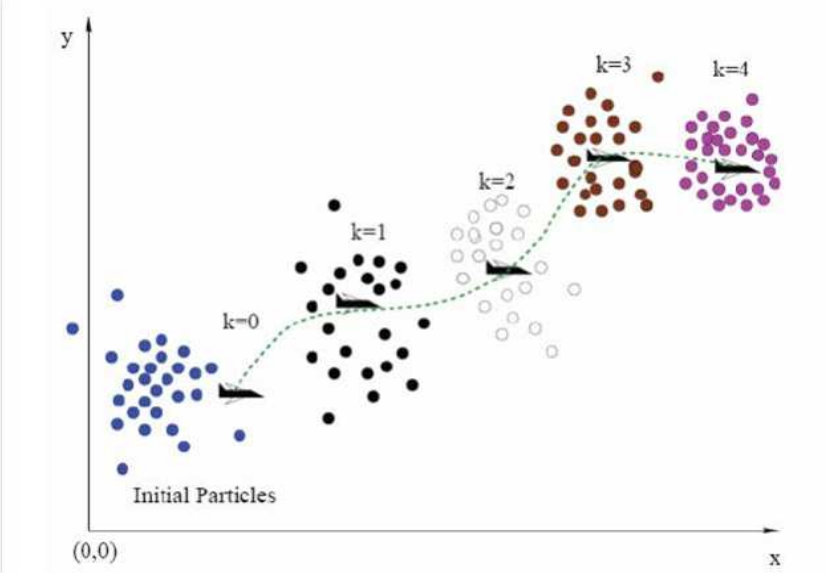


Figura 4.4: Convergencia de un filtro de partículas

4.4. Filtro de Partículas Distribuido

La carga computacional que conlleva la implementación de filtros de partículas provoca que en principio resulte poco atractivo para su utilización en aplicaciones reales. Sin embargo, el gran potencial de los filtros de partículas para la resolución de problemas de estimación hace que aparezca la idea de utilizar esquemas de implementación distribuidos para el procesado de los datos.

En esta sección se presentan de forma general algunos esquemas para la distribución de la carga computacional en filtros de partículas [12]. En estos esquemas distribuidos llamaremos *nodos* a todos aquellos sensores de la red que realizan la computación de los algoritmos, existiendo un único nodo destino, al que denominaremos *sink*, encargado de la fusión de toda la información y así como de dirigir la computación de la totalidad de la red.

Podemos distinguir principalmente entre tres tipos de filtros de partículas distribuidos [12]:

1. **Filtro de Partículas Distribuido Global** - *Global Distributed Particle Filter (GDPF)* -: Es el esquema más simple; tanto las partículas como los pesos son generados de forma local en cada uno de los nodos. La normalización de los pesos y el remuestreo queda a cargo del nodo *sink*, que debe enviar las partículas remuestreadas a la red completa. Este último también es el responsable de proporcionar la estimación final.

2. **Filtro de Partículas Distribuido Local** - *Local Distributed Particle Filter (LDPF)* - : Las partículas y los pesos asociados son calculados y normalizados de forma local. El remuestreo se realiza también de forma local, eliminando la necesidad del nodo sink de enviar las partículas de vuelta a los nodos. Cada nodo envía únicamente la información suficiente para la estimación de x_k, realizada por el sink.

3. **Filtro de partículas Distribuido Comprimido** - *Compressed Distributed Particle Filter (CDPF)* -: Al igual que en los esquemas anteriores las partículas y los pesos son calculadas de forma local en los nodos de la red. La normalización de los pesos y el remuestreo queda a cargo del nodo sink. Sin embargo, cada nodo envía únicamente una distribución reducida de las partículas lo suficientemente representativa como para poder realizar la estimación en destino. Se envía así la información de forma comprimida.

Los esquemas que han sido objeto de estudio en este Proyecto de Fin de Carrera y que se presentan en el capítulo siguiente se pueden englobar dentro del primer grupo de filtros de partículas distribuidos, GDPF. Por dicho motivo se muestra a continuación un esquema del funcionamiento general del algoritmo que lo implementa [12].

4.4.1. Esquema Algoritmo GDPF

Teniendo en cuenta la siguiente notación para un instante k:

- $\tilde{\mathbf{X}}_k = \left\{\tilde{\mathbf{x}}_k^{(m)}\right\}$ - Conjunto de partículas antes del remuestreo.
- $\tilde{W}_k = \left\{\tilde{w}_k^{(m)}\right\}$ - Conjunto de pesos no normalizados.
- $\mathbf{X}_k = \left\{\mathbf{x}_k^{(m)}\right\}$ - Conjunto de partículas remuestreadas.
- $W_k = \left\{w_k^{(m)}\right\}$ - Conjunto de pesos normalizados.
- n el nodo n-ésimo de la red, siendo $\{1, \ldots, N\}$ el conjunto de nodos de la red.
- M el número de partículas, y M_n el número de partículas del nodo n.

El algoritmo GDPF puede resumirse como:

1. En el instante k el *nodo sink* particiona el conjunto de todas las partículas del instante anterior $k-1$ en N subconjuntos de partículas $\tilde{\mathbf{X}}_{k-1}^n$, $n = 1, \cdots, N$

$$\mathbf{X}_{k-1} = \bigcup_{n=1}^{N} \mathbf{X}_{k-1}^n \Longrightarrow \mathbf{X}_{k-1}^n = \left\{\mathbf{x}_{k-1}^{n,(m)}\right\}_{m=1}^{M_n} \tag{4.28}$$

2. El *nodo sink* envía $\left\{\mathbf{X}_{k-1}^n, \mathbf{z}_k\right\}$ a cada nodo n, $n = 1, \cdots, N$
3. Cada nodo $n = 1, \cdots, N$ realiza en paralelo:
   - Utiliza $\mathbf{X}_{k-1}^n$ para generar las nuevas partículas sin remuestrearlpara el instante k, $\tilde{\mathbf{X}}_k^n$, según lo establecido en esquema del filtro de partículas, Sección 4.3.
   - Obtiene los pesos asociados a esas partículas, $\tilde{W}_k^n$ según la Sección 4.3.
   - Envía la información, $\tilde{\mathbf{X}}_k^n$ y $\tilde{W}_k^n$ al nodo destino.
4. El nodo destino, o *nodo sink*:
   - Normaliza y une los pesos recibidos $\tilde{W}_k = \bigcup_{n=1}^{N} \tilde{W}_k^n$ para obtener W_k.
   - Realiza el remuestreo de las partículas recibidas $\tilde{\mathbf{X}}_k = \bigcup_{n=1}^{N} \tilde{\mathbf{X}}_k^n$ para obtener $\mathbf{X}_k$.
   - Determina la estimación de la posición para el instante k, $\hat{\mathbf{x}}_k$.

Uno de los problemas de esta implementación, con el que trataremos en el próximo capítulo es la gran cantidad de datos que han de ser enviados a través de la red, tanto hacia el nodo destino, como del nodo sink al resto de nodos de la red, además de la gran carga computacional del nodo sink. En el siguiente capítulo se desarrollan una serie de esquemas de algoritmos de filtros de partículas distribuidos que derivan del expuesto aquí y que intentan dar solución a ambos problemas.

Capítulo 5

ESQUEMAS PROPUESTOS

En este capítulo se propone un esquema general de filtros de partículas distribuidos objeto de estudio y que buscan dar solución a los problemas de este tipo de filtros de partículas mencionados en el capítulo anterior. Además se proponen distintas variantes a dicho esquema.

Se describen además otras técnicas aplicadas en el modelo de estudio para posibilitar un mejor comportamiento del esquema general, como la aplicación de fusión de datos en los paquetes de estimación o la modificación de la estructura de enrutado.

5.1. Esquema Propuesto

La razón principal de distribuir la carga computacional que tiene lugar en el filtrado de partículas viene de la necesidad de utilizar todas las partículas para obtener la estimación y el remuestreo para la próxima iteración [12]. En este caso aparecen tres clases de nodos que intervienen en el algoritmo distribuido:

1. Los *nodos* de que conforman la red. Se encuentran distribuidos de forma más o menos uniforme en el área de acción de la red de sensores y realizan tareas de sensado y/o enrutado. La red completa se divide en grupos de nodos o *clusters*.

2. Los *cluster-heads*, o líderes de cada uno de los *clusters*. Existe uno por cada *cluster* y son nodos encargados de realizar el procesado computacional además de las tareas de un nodo normal.

3. El nodo destino o *sink node*, encargado de dirigir la computación del algoritmo, de realizar la estimación final y determinar si es necesario o no un proceso de remuestreo.

Para un uso más eficiente de todos los nodos que conforman la red, el nodo *sink* no sólo realiza las funciones de destino, sino que también actúa como *cluster-head* del *cluster* al que pertenece.

En la modificación del algoritmo global de filtro de partículas distribuido, GDPF, que se propone en este estudio tanto las partículas como los pesos asociados a estas se calculan de forma local en los *cluster-heads*, mientras que las tareas de estimación y remuestreo son realizadas por el *nodo sink* que posee una visión global de la red.

Dentro del modelo objeto de estudio, tenemos C filtros de partículas corriendo en cada uno de los C *cluster-heads* de la red. Se considera en este caso que los filtros de partículas están *sincronizados*. Dos filtros de partículas están sincronizados si las trazas aleatorias que generan son siempre las mismas, garantizando así que todos ellos trabajarán con el mismo conjunto de partículas.

La función de verosimilitud determinada de forma local por cada uno de los *cluster-heads* es enviada al nodo destino - $p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right)$ con $c = 1, \cdots, C$ siendo C el número de *clusters* de la red -. El *sink* es responsable de realizar la estimación final, así como enviar las partículas remuestreadas de vuelta a los *cluster-heads* de la red [12], pues posee la información completa en el instante k, esto es, la función de verosimilitud de la red en ese instante, $p\left(\mathbf{z}_k|\mathbf{x}_k^{(m)}\right)$.

Este filtro de partículas funciona de la siguiente manera (Figura 5.1):

Todos los *cluster-head* están sincronizados: parten del mismo conjunto de partículas y sus correspondientes pesos $\left\{\mathbf{x}_{k-1}^{(m)}, w_{k-1}^{(m)}\right\}$.

$$\begin{aligned} \mathbf{x}_{k-1}^{(m)}: &\quad \mathbf{x}_{k-1}^{(m)} = \left[x_{1,k-1}^{(m)},\, x_{2,k-1}^{(m)},\, \dot{x}_{1,k-1}^{(m)},\, \dot{x}_{2,k-1}^{(m)}\right] \quad m = 1, 2, \ldots, M \\ w_{k-1}^{(m)}: &\quad normalizados \quad \sum_{m=1}^{M} w_{k-1}^{(m)} = 1 \end{aligned} \tag{5.1}$$

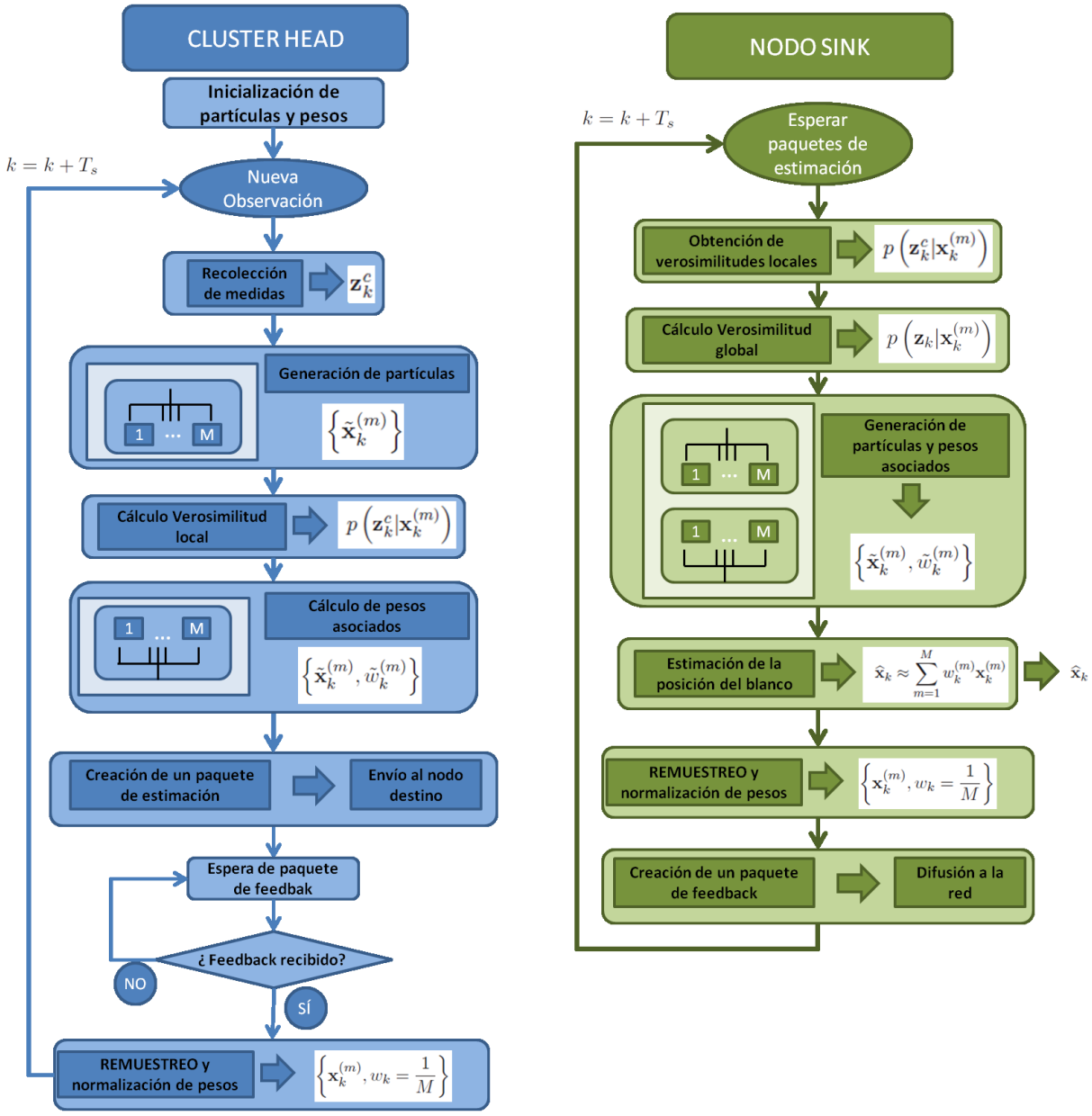


Figura 5.1: Esquema general de funcionamiento del algoritmo del filtro de partículas distribuido

En cada *cluster-head* se realizan de forma local los siguientes pasos para cada instante de muestreo k:

1. **Recolección de medidas**: Recolecta las medidas realizadas por los miembros de su *cluster* con las que formará el vector $\mathbf{z}_k^c$ donde $c = \{1, \cdots, C\}$.

2. **Generación de partículas**: Calcula las nuevas partículas sin remuestrear $\left\{\tilde{\mathbf{x}}_k^{(m)}\right\}$.

3. **Cálculo de la verosimilitud local**: Determina la función de verosimilitud local con la información que tiene disponible, $p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right)$.

4. **Cálculo de los pesos asociados**: de las partículas sin remuestrear $\left\{\tilde{\mathbf{x}}_k^{(m)}, \tilde{w}_k^{(m)}\right\}$.

5. **Creación y envío de un paquete de estimación**: Genera un paquete con la similitud local previamente calculada, $p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right)$, que envía hacia el nodo destino.

6. **Espera de información de remuestreo**: Permanece a la espera hasta recibir por parte del *sink* la información del remuestreo que le indicará cuál es el nuevo conjunto de partículas con el que deberá continuar el algoritmo en la siguiente iteración.

$$\left\{\mathbf{x}_k^{(m)}, w_k^{(m)}\right\}, \quad con \quad w_k^{(m)} = 1/M. \tag{5.2}$$

7. Establece $k = k + T_s$, siendo T_s el periodo de muestreo o *sampling*, y vuelve al paso 1.

El nodo *sink* realiza los siguientes pasos:

1. **Recepción de paquetes de estimación**: Recibe las informaciones parciales de la función de verosimilitud y construye la función de verosimilitud global $p\left(\mathbf{z}_k|\mathbf{x}_k\right)$, que contiene la información completa de la red en el instante k en las que fueron obtenidas las observaciones utilizadas para el cálculo de las verosimilitudes locales.

$$p\left(\mathbf{z}_k|\mathbf{x}_k^{(m)}\right) = \prod_{c=1}^{C} p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right) \tag{5.3}$$

2. **Generación de partículas y pesos asociados**: Genera las nuevas partículas antes del remuestreo y los pesos asociados, normalizándolos $\left(\sum_{m=1}^{M} \tilde{w}_k^{(m)} = 1\right)$:

$$\tilde{w}_{k-1}^{(m)} \rightarrow \tilde{w}_k^{(m)} \tag{5.4}$$

$$\tilde{w}_k^{(m)} = \tilde{w}_{k-1}^{(m)} p\left(\mathbf{z}_k|\tilde{\mathbf{x}}_k^{(m)}\right) \tag{5.5}$$

3. **Estimación de la posición del blanco en el instante** k: Realiza la estimación de la posición a través de la minimización del error cuadrático medio *MMSE*

$$\begin{aligned} \widehat{\mathbf{x}}_k &= E\left[\mathbf{x}_k|\overline{\mathbf{z}_k}\right] \\ &= \int \mathbf{x}_k p\left(\mathbf{x}_k|\overline{\mathbf{z}_k}\right) d\mathbf{x}_k \end{aligned} \tag{5.6}$$

aproximando (5.24) como

$$\widehat{\mathbf{x}}_k \approx \sum_{m=1}^{M} w_k^{(m)} \mathbf{x}_k^{(m)} \tag{5.7}$$

4. **Remuestreo**: Realiza el remuestreo, replicando aquellas partículas de mayor importancia y eliminando las que aporten menos información

$$\left\{ \mathbf{x}_k^{(m)}, w_k = \frac{1}{M} \right\}$$

5. **Creación y envío de un paquete de retroalimentación o *feedback***: Se envía a todos los sensores de la red la información del remuestreo: qué partículas han sido replicadas y en qué número.

6. Establece internamente $k = k + T_s$ y vuelve al paso 1.

Uno de los problemas más obvios que surgen en este tipo de implementaciones es la gran cantidad de datos que han de ser enviados a través de la red, tanto desde cada uno de los *cluster-heads* hacia el nodo *sink* como desde este último a los diferentes nodos de la red.

En las siguientes secciones se plantean soluciones a ambos problemas, introduciendo ciertas variaciones para estudiar un posible aumento de la vida de la red. Se intentará en ellos disminuir la cantidad de información enrutada, ya que es este proceso el que mayor gasto de energía conlleva.

5.2. Modelo 1

En este modelo todos los sensores que actúan como *cluster-heads* dentro de la red poseen una misma constante, que denominaremos *irm* o *instantes para el remuestreo*. *irm* indica el número de periodos de muestreo Ts (*Time of Sampling*) que han de ejecutar el filtro de partículas sin realizar remuestreo antes de determinar que este es necesario.

Cada *cluster-head* posee un contador propio, irm_c, en el que almacena el número de iteraciones en las que no se ha realizado remuestreo en el filtro de partículas (Figura 5.2).

1. **Recolección de medidas**: Recolecta las medidas realizadas por los miembros de su *cluster* con las que formará el vector $\mathbf{z}_k^c$ donde $c = \{1, \cdots, C\}$.

2. **Generación de partículas**: Calcula las nuevas partículas sin remuestrear $\left\{\tilde{\mathbf{x}}_k^{(m)}\right\}$.

3. **Cálculo de la verosimilitud local**: Determina la función de verosimilitud local con la información que tiene disponible, $p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right)$.

4. **Cálculo de los pesos asociados**: de las partículas sin remuestrear $\left\{\tilde{\mathbf{x}}_k^{(m)}, \tilde{w}_k^{(m)}\right\}$.

5. **Creación y envío de un paquete de estimación**: Genera un paquete con la similitud local previamente calculada, $p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right)$, que envía hacia el nodo destino.

6. **Comparación del contador local de instantes para el remuestreo**: Comprueba si se cumple la condición $irm_c \equiv irm$:

   - Si $irm_c \neq irm$, el *cluster-head* no realizará el paso de remuestreo.

     a) Aumenta su contador $irm_c = irm_c + 1$.

   - Si $irm_c \equiv irm$, el *cluster-head* realizará remuestreo

     a) Reinicia su contador $irm_c = 0$.

     b) **Espera de información de remuestreo**: Permanece a la espera hasta recibir por parte del *sink* la información del remuestreo que le indicará cuál es el nuevo conjunto de partículas con el que deberá continuar el algoritmo en la siguiente iteración

     $$\left\{\mathbf{x}_k^{(m)}, w_k^{(m)}\right\}, \quad con \quad w_k^{(m)} = 1/M. \tag{5.8}$$

7. Toma $k = k + T_s$ iniciando la siguiente iteración del algoritmo.

El nodo *sink* posee también la misma constante irm y un contador propio irm_S en el que almacena el número de iteraciones en las que no se ha realizado remuestreo. En cada iteración comprueba la condición $irm_S \equiv irm$:

1. **Recepción de paquetes de estimación**: Recibe las informaciones parciales de la función de verosimilitud, y, construye la función de verosimilitud global $p\left(\mathbf{z}_k|\mathbf{x}_k\right)$, que contiene la información completa de la red en el instante k en las que fueron obtenidas las observaciones utilizadas para el cálculo de las verosimilitudes locales.

$$p\left(\mathbf{z}_k|\mathbf{x}_k^{(m)}\right) = \prod_{c=1}^{C} p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right) \tag{5.9}$$

2. **Generación de partículas y pesos asociados**: Genera las nuevas partículas antes del remuestreo y los pesos asociados, normalizándolos $\left(\sum_{m=1}^{M} \tilde{w}_k^{(m)} = 1\right)$:

$$\tilde{w}_{k-1}^{(m)} \rightarrow \tilde{w}_k^{(m)} \tag{5.10}$$

$$\tilde{w}_k^{(m)} = \tilde{w}_{k-1}^{(m)} p\left(\mathbf{z}_k | \tilde{\mathbf{x}}_k^{(m)}\right) \tag{5.11}$$

3. **Estimación de la posición del blanco en el instante** k: Realiza la estimación de la posición a través de la minimización del error cuadrático medio *MMSE*

$$\begin{aligned} \widehat{\mathbf{x}}_k &= E\left[\mathbf{x}_k | \overline{\mathbf{z}_k}\right] \\ &= \int \mathbf{x}_k p\left(\mathbf{x}_k | \overline{\mathbf{z}_k}\right) d\mathbf{x}_k \end{aligned} \tag{5.12}$$

aproximando (5.24) como

$$\widehat{\mathbf{x}}_k \approx \sum_{m=1}^{M} w_k^{(m)} \mathbf{x}_k^{(m)} \tag{5.13}$$

4. **Comparación del contador local de instantes para el remuestreo**: Comprueba si se cumple la condición $irm_S \equiv irm$:

   - Si $irm_S \neq irm$ la retroalimentación al resto de nodos de la red no resulta necesaria, por lo que el nodo destino no deberá construir ni difundir ningún mensaje de *feedback*.

     a) Aumenta su contador $irm_S = irm_S + 1$.

   - Si $irm_S \equiv irm$, ha de realizar remuestreo y difundir la información a la red

     a) Reinicializa el contador, $irm_S = 0$

     b) **Remuestreo**: Realiza el remuestreo, replicando aquellas partículas de mayor importancia y eliminando las que aporten menos información

$$\left\{\mathbf{x}_k^{(m)}, w_k = \frac{1}{M}\right\}$$

     c) **Creación y envío de un paquete de retroalimentación o *feedback***: Se envía a todos los sensores de la red la información del remuestreo: qué partículas han sido replicadas y en qué número.

5. Actualiza $k = k + T_s$ iniciando la siguiente iteración del algoritmo.

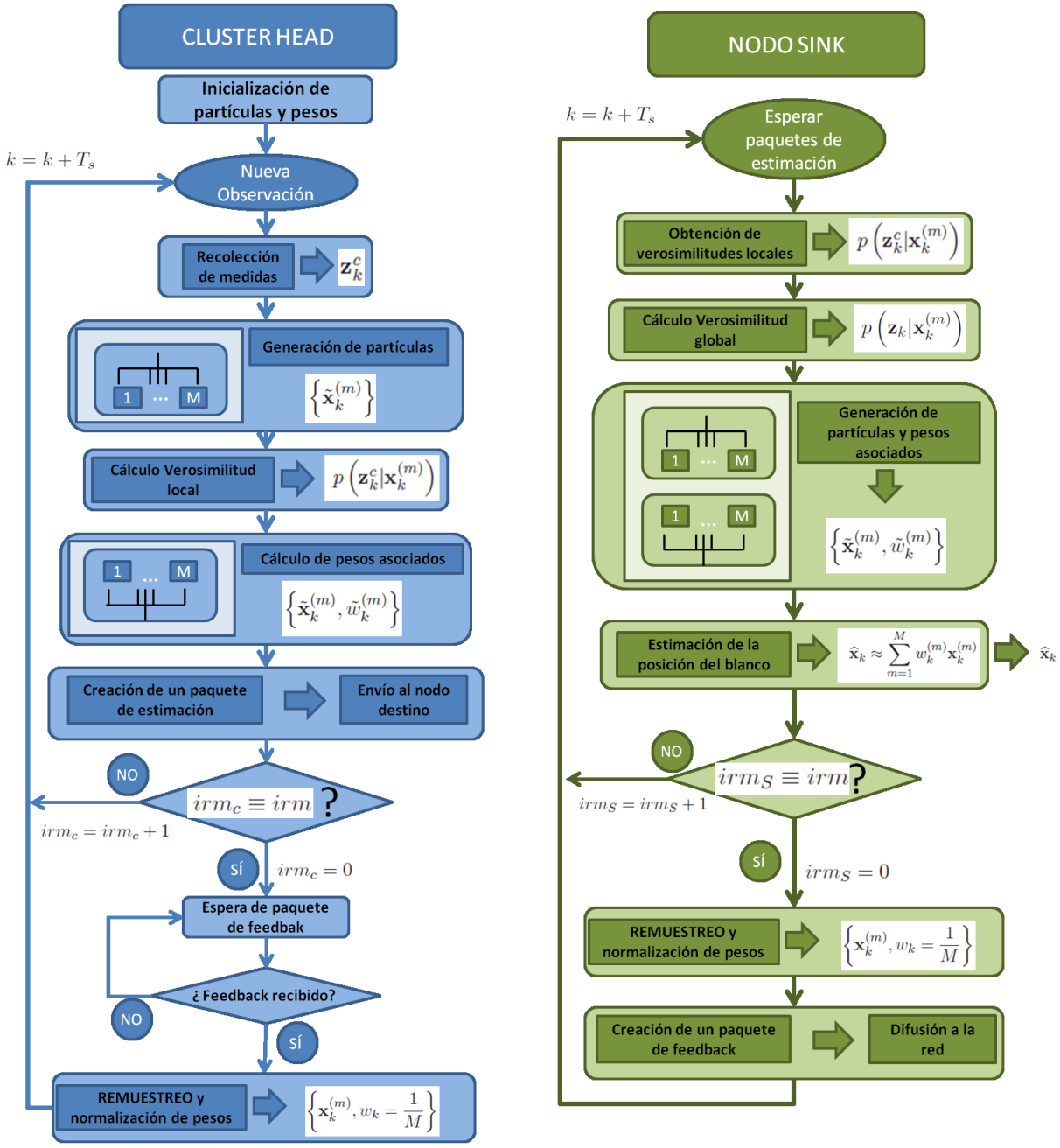


Figura 5.2: Esquema de funcionamiento del algoritmo del filtro de partículas distribuido del Modelo 1

Se ha de notar que los contadores de cada uno de los nodos de la red se encuentran sincronizados, pues en la primera iteración todos ellos se inicializan a 0.

Utilizando esta variable se posibilita el aumento de la vida de la red respecto al esquema

general propuesto en la Sección 5.1 pues la resincronización de todas las partículas de la red en cada uno de los *cluster-heads* se realiza únicamente cada cierto tiempo, posibilitando la convergencia del algoritmo de filtro de partículas y disminuyendo el error producido en la estimación del blanco en cada instante k. El comportamiento de ambos modelos coincidirá en caso de que la variable *irm* se fije a 0.

5.3. Modelo 2

En este modelo todos los *cluster-heads* de la red utilizan el concepto de partículas eficaces, N_{eff}, definido en la Sección 4.2.1, ya que es un claro indicador de la degeneración del filtro de partículas.

Cada uno de los *cluster-heads* calcula en cada iteración del algoritmo del filtro de partículas el número de partículas eficaces correspondiente N_{eff}^c, con $c = 1 \cdots C$ siendo C el número de *clusters* de la red.

En cada paquete de estimación, cada *cluster-head* activará o no un flag específico que indicará su necesidad de remuestreo. Es el nodo *sink* el que determina, una vez ha recibido toda la información, si el remuestreo resulta necesario. Esto es, si el número de *cluster-heads* que necesitan remuestreo supera un umbral previamente fijado.

El esquema de este modelo se puede resumir en los siguientes pasos (Figura 5.3):
Cada *cluster-head*:

1. **Recolección de medidas**: Recolecta las medidas realizadas por los miembros de su *cluster* con las que formará el vector $\mathbf{z}_k^c$ donde $c = \{1, \cdots, C\}$.

2. **Generación de partículas**: Calcula las nuevas partículas sin remuestrear $\left\{\tilde{\mathbf{x}}_k^{(m)}\right\}$.

3. **Cálculo de la verosimilitud local**: Determina la función de verosimilitud local con la información que tiene disponible, $p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right)$.

4. **Cálculo de los pesos asociados**: de las partículas sin remuestrear $\left\{\tilde{\mathbf{x}}_k^{(m)}, \tilde{w}_k^{(m)}\right\}$.

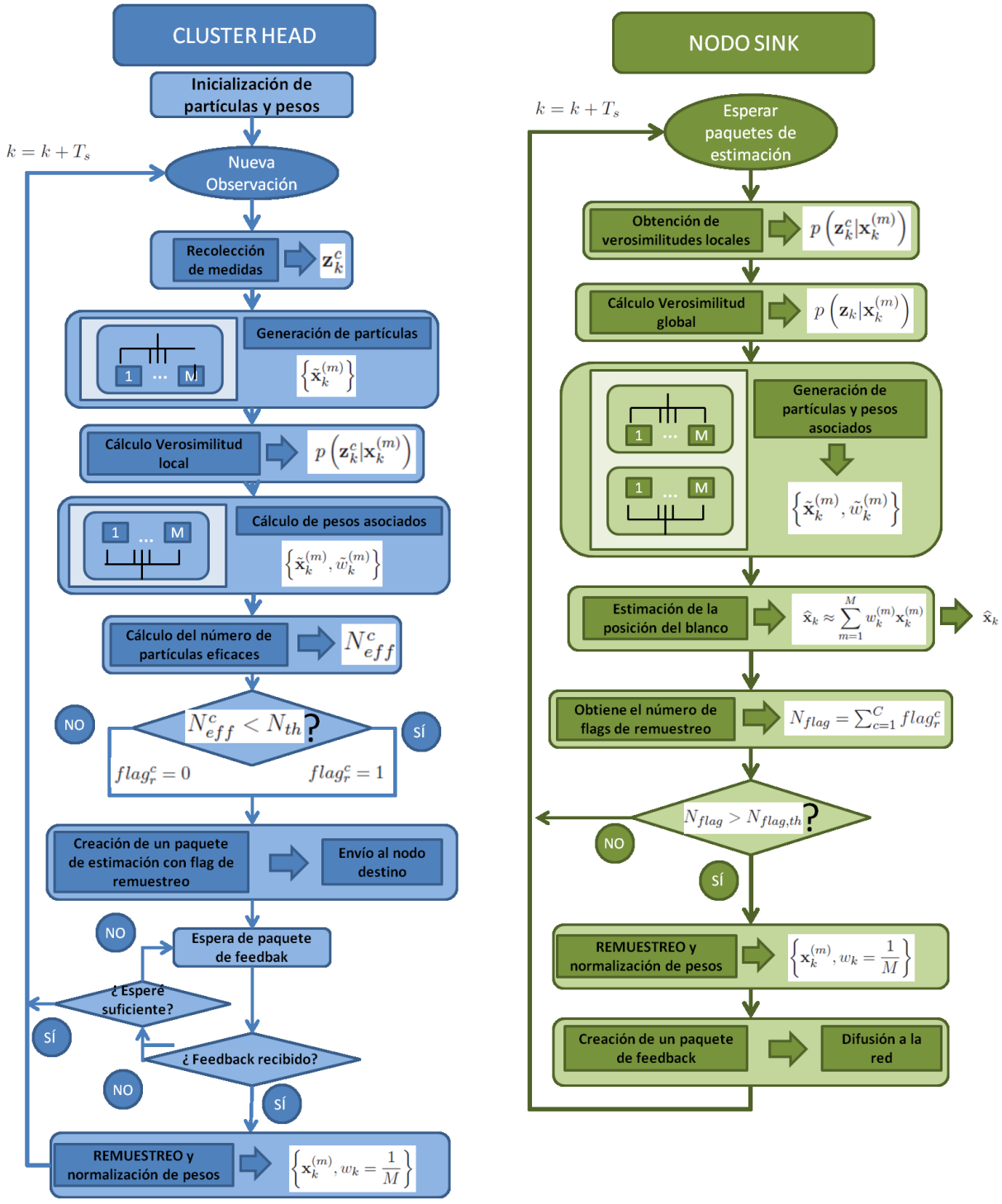


Figura 5.3: Esquema de funcionamiento del algoritmo del filtro de partículas distribuido del Modelo 2

5. **Cálculo del número de partículas eficaces**: Calcula el número de partículas eficaces N_{eff}^c y lo compara con un umbral N_{th}.

   - Si $N_{eff}^c < N_{th}$, determina que el remuestreo es necesario

     a) Activa el flag de remuestreo, $flag_r^c = 1$

   - Si $N_{eff}^c \geq N_{th}$, el remuestreo no es necesario

     a) Coloca el flag de remuestreo a 0 $flag_r^c = 0$

6. **Creación y envío de un paquete de estimación**: Genera un paquete con la similitud local previamente calculada, $p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right)$, añadiendo el flag de remuestreo y lo envía hacia el nodo destino.

7. **Espera de información de remuestreo p instantes**: Permanece a la espera hasta recibir por parte del *sink* la información del remuestreo que le indicará cuál es el nuevo conjunto de partículas con el que deberá continuar el algoritmo en la siguiente iteración.

   - Si en p instantes recibe el mensaje de feedback, realiza el remuestreo.

   $$\left\{\mathbf{x}_k^{(m)}, w_k^{(m)}\right\}, \quad con \quad w_k^{(m)} = 1/M. \tag{5.14}$$

   - Si en p instantes no ha recibido el paquete con los índices de remuestreo, considera que este no es necesario.

8. Toma $k = k + T_s$ iniciando la siguiente iteración del algoritmo.

El nodo *sink*:

1. **Recepción de paquetes de estimación**: Recibe las informaciones parciales de la función de verosimilitud, y, construye la función de verosimilitud global $p\left(\mathbf{z}_k|\mathbf{x}_k\right)$, que contiene la información completa de la red en el instante k en las que fueron obtenidas las observaciones utilizadas para el cálculo de las verosimilitudes locales.

$$p\left(\mathbf{z}_k|\mathbf{x}_k^{(m)}\right) = \prod_{c=1}^{C} p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right) \tag{5.15}$$

2. **Generación de partículas y pesos asociados**: Genera las nuevas partículas antes del remuestreo y los pesos asociados, normalizándolos $\left(\sum_{m=1}^{M} \tilde{w}_k^{(m)} = 1\right)$:

$$\tilde{w}_{k-1}^{(m)} \rightarrow \tilde{w}_k^{(m)} \tag{5.16}$$

$$\tilde{w}_k^{(m)} = \tilde{w}_{k-1}^{(m)} p\left(\mathbf{z}_k | \tilde{\mathbf{x}}_k^{(m)}\right) \tag{5.17}$$

3. **Estimación de la posición del blanco en el instante** k: Realiza la estimación de la posición a través de la minimización del error cuadrático medio *MMSE*

$$\begin{aligned} \widehat{\mathbf{x}}_k &= E\left[\mathbf{x}_k | \overline{\mathbf{z}_k}\right] \\ &= \int \mathbf{x}_k p\left(\mathbf{x}_k | \overline{\mathbf{z}_k}\right) d\mathbf{x}_k \end{aligned} \tag{5.18}$$

aproximando (5.24) como

$$\widehat{\mathbf{x}}_k \approx \sum_{m=1}^{M} w_k^{(m)} \mathbf{x}_k^{(m)} \tag{5.19}$$

4. **Decisión de remuestreo**: Extrae de cada paquete la información del flag de remuestreo, sumando para obtener el número de *cluster-heads* cuyo algoritmo se ha degenerado lo suficiente para necesitarlo $N_{flag} = \sum_{c=1}^{C} flag_r^c$

   - Si $N_{flag} > N_{flag,th}$ decide que el remuestreo es necesario:

     a) **Remuestreo**: Realiza el remuestreo, replicando aquellas partículas de mayor importancia y eliminando las que aporten menos información

     $$\left\{\mathbf{x}_k^{(m)}, w_k = \frac{1}{M}\right\}$$

     b) **Creación y envío de un paquete de retroalimentación o *feedback***: Se envía a todos los sensores de la red la información del remuestreo: qué partículas han sido replicadas y en qué número.

   - Si $N_{flag} \leq N_{flag,th}$, el remuestreo no es necesario

5. Actualiza $k = k + T_s$ iniciando la siguiente iteración del algoritmo.

5.4. Modelo 3

Este modelo es similar al Modelo 2, pero con una modificación en la toma de decisiones de remuestreo para intentar optimizar el funcionamiento del algoritmo.

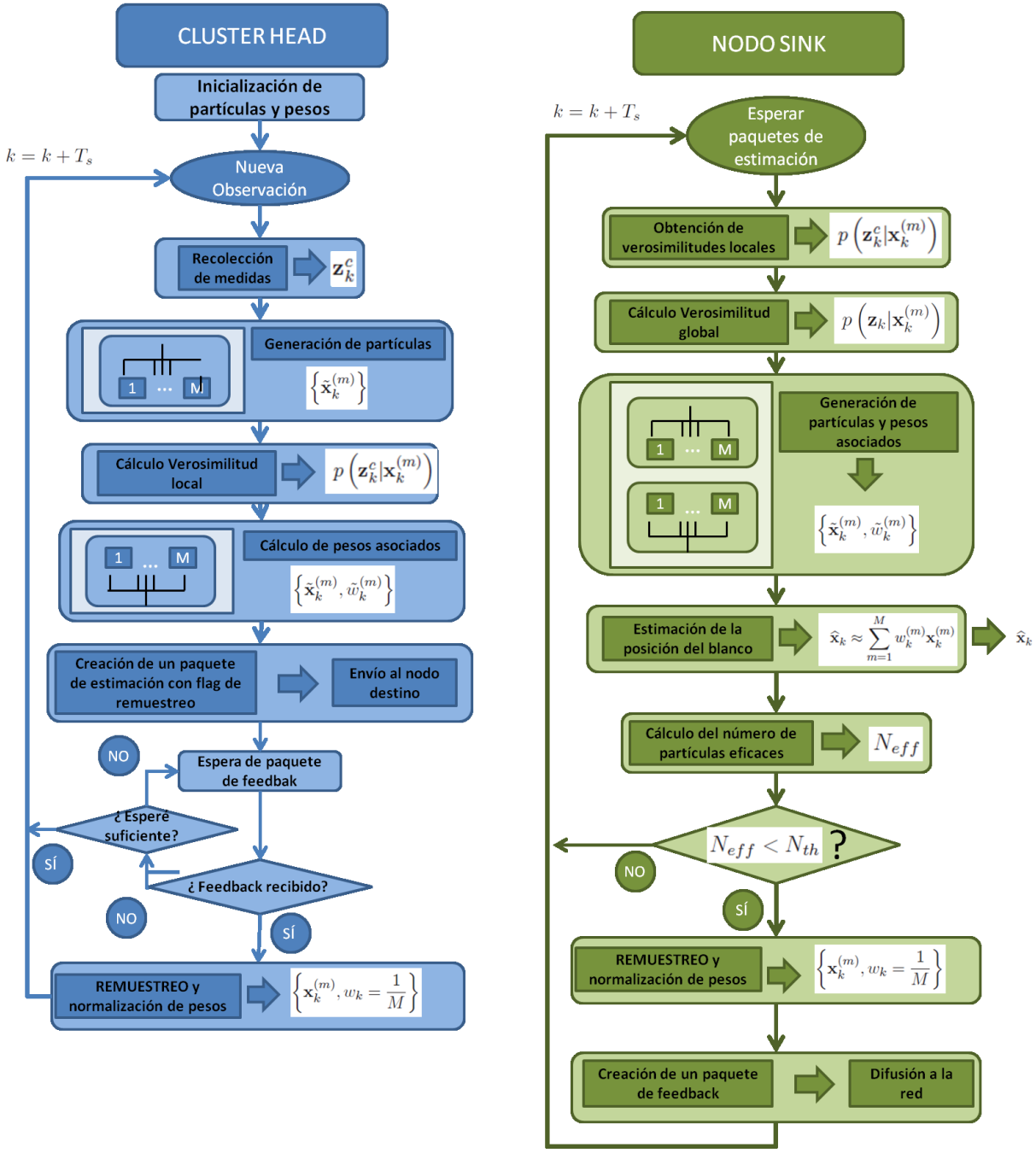


Figura 5.4: Esquema de funcionamiento del algoritmo del filtro de partículas distribuido del Modelo 3

Aquí todos los *cluster-heads* de la red ejecutan el filtro de partículas de forma distribuida. No calculan el número de partículas eficaces N_{eff} en cada instante sino que envían únicamente la función de verosimilitud local que han calculado al nodo sink. Una vez realizada esta parte del

algoritmo se colocan en modo espera. En el nodo Sink, una vez recibida la información de cada uno de los nodos y calculada la función de verosimilitud completa se calcula el número de partículas eficaces N_{eff}, comparándolo con un umbral determinado y decidiendo si se remuestrea o no.

Los pasos a seguir por cada uno de los nodos serían (Figura 5.4):
Cada *cluster-head*:

1. **Recolección de medidas**: Recolecta las medidas realizadas por los miembros de su *cluster* con las que formará el vector $\mathbf{z}_k^c$ donde $c = \{1, \cdots, C\}$.
2. **Generación de partículas**: Calcula las nuevas partículas sin remuestrear $\left\{\tilde{\mathbf{x}}_k^{(m)}\right\}$.
3. **Cálculo de la verosimilitud local**: Determina la función de verosimilitud local con la información que tiene disponible, $p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right)$.
4. **Cálculo de los pesos asociados**: de las partículas sin remuestrear $\left\{\tilde{\mathbf{x}}_k^{(m)}, \tilde{w}_k^{(m)}\right\}$.
5. **Creación y envío de un paquete de estimación**: Genera un paquete con la similitud local previamente calculada, $p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right)$, que envía hacia el nodo destino.
6. **Espera de información de remuestreo p instantes**: Permanece a la espera hasta recibir por parte del *sink* la información del remuestreo que le indicará cuál es el nuevo conjunto de partículas con el que deberá continuar el algoritmo en la siguiente iteración.
   - Si en p instantes recibe el mensaje de feedback, realiza el remuestreo.
   $$\left\{\mathbf{x}_k^{(m)}, w_k^{(m)}\right\}, \quad con \quad w_k^{(m)} = 1/M. \tag{5.20}$$
   - Si en p instantes no ha recibido el paquete con los índices de remuestreo, considera que este no es necesario.
7. Toma $k = k + T_s$ iniciando la siguiente iteración del algoritmo.

El nodo *sink*:

1. **Recepción de paquetes de estimación**: Recibe las informaciones parciales de la función de verosimilitud y construye la función de verosimilitud global $p\left(\mathbf{z}_k|\mathbf{x}_k\right)$, que contiene la

información completa de la red en el instante k en las que fueron obtenidas las observaciones utilizadas para el cálculo de las verosimilitudes locales.

$$p\left(\mathbf{z}_k|\mathbf{x}_k^{(m)}\right) = \prod_{c=1}^{C} p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right) \tag{5.21}$$

2. **Generación de partículas y pesos asociados**: Genera las nuevas partículas antes del remuestreo y los pesos asociados, normalizándolos $\left(\sum_{m=1}^{M} \tilde{w}_k^{(m)} = 1\right)$:

$$\tilde{w}_{k-1}^{(m)} \rightarrow \tilde{w}_k^{(m)} \tag{5.22}$$

$$\tilde{w}_k^{(m)} = \tilde{w}_{k-1}^{(m)} p\left(\mathbf{z}_k|\tilde{\mathbf{x}}_k^{(m)}\right) \tag{5.23}$$

3. **Estimación de la posición del blanco en el instante** k: Realiza la estimación de la posición a través de la minimización del error cuadrático medio *MMSE*

$$\begin{aligned} \widehat{\mathbf{x}}_k &= E\left[\mathbf{x}_k|\overline{\mathbf{z}_k}\right] \\ &= \int \mathbf{x}_k p\left(\mathbf{x}_k|\overline{\mathbf{z}_k}\right) d\mathbf{x}_k \end{aligned} \tag{5.24}$$

aproximando (5.24) como

$$\widehat{\mathbf{x}}_k \approx \sum_{m=1}^{M} w_k^{(m)} \mathbf{x}_k^{(m)} \tag{5.25}$$

4. **Cálculo del número de partículas eficaces y decisión de remuestreo**: Calcula en número de partículas eficaces N_{eff} de las nuevas partículas y lo compara con un umbral determinado N_{th}.

   - Si $N_{eff} < N_{th}$ decide que el remuestreo es necesario:

     a) **Remuestreo**: Realiza el remuestreo, replicando aquellas partículas de mayor importancia y eliminando las que aporten menos información

     $$\left\{\mathbf{x}_k^{(m)}, w_k = \frac{1}{M}\right\}$$

     b) **Creación y envío de un paquete de retroalimentación o *feedback***: Se envía a todos los sensores de la red la información del remuestreo: qué partículas han sido replicadas y en qué número.

   - Si $N_{eff} \geq N_{th}$, el remuestreo no es necesario

5. Actualiza $k = k + T_s$ iniciando la siguiente iteración del algoritmo.

5.5. Umbral variable del número de partículas eficaces

En los Modelos 2 y 3, aquellos en los que se calcula el número de partículas eficaces, se observó que el tiempo de convergencia tras la inicialización del algoritmo aumentaba con respecto al Modelo 1. Se decidió para intentar disminuir este tiempo de convergencia introducir un umbral variable para decidir la necesidad o no de remuestreo en cada instante k. Esta función $N_{th}(k)$ es una función asintótica definida por la siguiente ecuación (Figura 5.5):

$$N_{th}\left(k\right) = \frac{N_{th_{mean}}}{1 + e^{(A+B\cdot k)}} + N_{th_{min}} \tag{5.26}$$

donde:

- $N_{th_{min}}$ es el umbral mínimo, que corresponderá con el umbral fijado en los casos anteriores, N_{th}
- $N_{th_{mean}} = \left(N_{th_{max}-N_{th_{min}}}\right) \cdot \left(1 + e^{A}\right)$
- $N_{th_{max}} = M$ es el umbral máximo, es decir, el número de partículas presente en cada uno de los filtros.
- A y B son constantes fijas que cumplen $A < 0$, $B > 0$

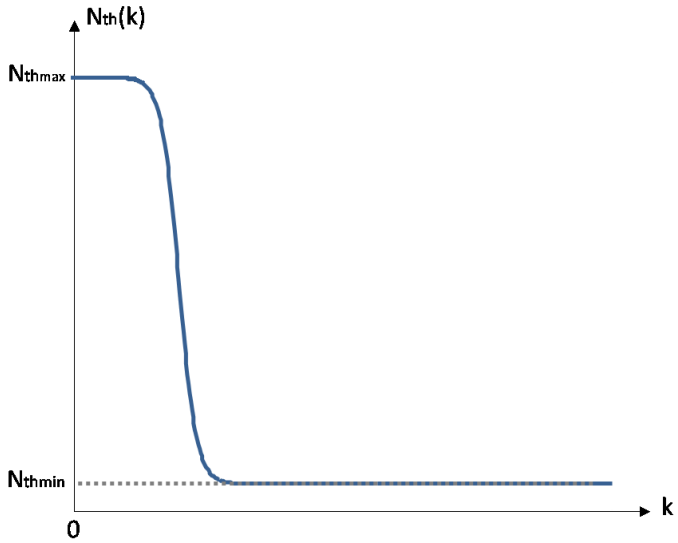


Figura 5.5: Función $N_{th}(k)$

Así, inicialmente nos encontraremos en el caso peor, en el que el umbral es igual al número de partículas $N_{th_{max}} = M$ del algoritmo, por lo que la decisión del nodo será siempre la de remuestrear. El valor de la función disminuye con k de forma continua hasta un valor mínimo $N_{th_{min}}$ que corresponde con el valor del umbral en el caso anterior, en el que no dependía del tiempo.

5.6. Tiempo de espera del mensaje de feedback variable

Al considerar que la red queda inactiva cuando el destino queda aislado, en los Modelos 2 y 3 resultaba necesario que los nodos esperaran un número elevado de instantes para recibir el mensaje de remuestreo. Cuando el número de nodos inactivos comienza a aumentar, aumenta también el tiempo de routing de los paquetes de estimación y, por tanto, el tiempo que el nodo destino necesitaba para decidir la necesidad de remuestreo. Si se consideraba un T_{wait} fijo en el tiempo este valor se alejaba mucho del valor óptimo en los primeros instantes de tiempo de vida de la red.

Por este motivo en los Modelos 2 y 3 se ha decidido que el tiempo máximo que un nodo debe esperar en cada instante T_s dependa del número de nodos inactivos en la red. Esto posibilita que, en caso de que el remuestreo no sea necesario y el nodo *sink* no envíe un mensaje de feedback en difusión, los nodos no tengan que esperar más del tiempo necesario. Esta función se define como $T_{wait}(N_{DeadSensors})$ a través de la siguiente ecuación (Figura 5.6).

$$f\left(N_{DeadSensors}\right) = \frac{T_{wait_{mean}}}{1 + e^{(A+B\cdot N_{DeadSensors})}} + T_{wait_{max}} \tag{5.27}$$

$$T_{wait}\left(N_{DeadSensors}\right) = \begin{cases} \lfloor (f\left(N_{DeadSensors}\right)) \rceil & N_{DeadSensors} > 0 \\ T_{wait_{min}} & N_{DeadSensors} \leq 0 \end{cases} \tag{5.28}$$

donde:

- $T_{wait_{mean}} = ((T_{wait_{min}} + 1) - T_{wait_{max}}) \cdot \left(1 + e^{A}\right)$
- $T_{wait_{min}}$ y $T_{wait_{max}}$ son constantes fijadas según la estructura y dimensiones de la red. Dependiendo del número máximo de saltos que un paquete de estimación tenga que dar para llegar al destino

- A y B son constantes fijadas de modo que cumplan que $A > 0$ y $0 < B < 1$.
- $\lfloor \cdot \rceil$ es la función de redondeo al entero más próximo.

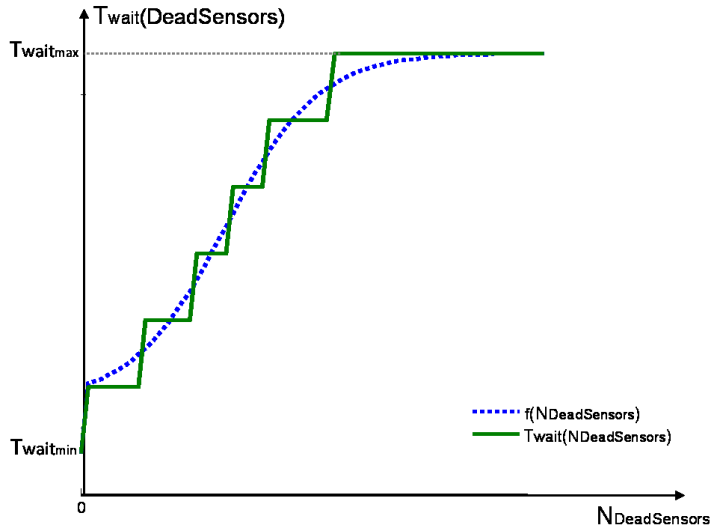


Figura 5.6: Función $T_{wait}(N_{DeadSensors})$

Es el nodo *sink* el encargado de difundir al resto de los nodos de la red por medio de un campo específico en el mensaje de *feedback* que deben aumentar el tiempo de espera de los siguientes mensajes de retroalimentación, T_{wait}, pues el nodo destino posee información completa de la red, incluyendo cuáles nodos han pasado a estar inactivos.

Inicialmente el tiempo de espera es mínimo, $T_{wait_{min}}$, pues todos los nodos de la red se encuentran activos. El tiempo de espera aumentará en números enteros consecutivos hasta llegar a un tiempo de espera máximo, $T_{wait_{max}}$. Ambas constantes serán fijadas según las dimensiones de la red para posibilitar una mayor eficiencia de esta, pues:

- Si $T_{wait_{max}} \uparrow\uparrow$, en caso de que el remuestreo no fuera necesario un nodo cualquiera de la red estaría esperando demasiado hasta percibir que el remuestreo no se realiza en esa iteración del filtro de partículas. En este caso el retardo existente entre la posición a tiempo real del blanco en movimiento y la última estimación realizada sería mayor.
- Si, por el contrario $T_{wait_{max}} \downarrow\downarrow$, puede ocurrir el caso en el que un mensaje de feedback llegue a un nodo demasiado tarde, y este haya pasado a la siguiente iteración sin llegar a

realizar remuestreo. Nos encontraríamos en el peor de los casos en el que no se remuestree nunca y el algoritmo del filtro de partículas acabe divergiendo debido al deterioro de la eficiencia del filtro, según lo explicado en la Sección 4.2.1.

5.7. Algoritmos de enrutado

Para todos y cada uno de los modelos presentados en las secciones anteriores se aplican dos tipos de algoritmo de enrutado. Uno completo, en el que todos los nodos que forman la red tienen sus capacidades de *routing* activadas. Y otro en el que los nodos que no actúan como *cluster-heads* tienen las capacidades de enrutado inactivas, y, sólo las activan en caso de que un *cluster-head* les pase el testigo de líder del *cluster* correspondiente:

- **Routing completo, salto a salto**: En este algoritmo intervienen todos los nodos de la red. Para el enrutado se considera que dos nodos son vecinos si se encuentran dentro de un radio de transmisión R, pertenezcan o no al mismo *cluster*. En el caso en el que la estructura física de la red tenga forma de rejilla, un nodo puede tener hasta 8 vecinos. Cada nodo tiene conocimiento no sólo de cuáles son sus vecinos, sino también de su localización. Tanto la comunicación como el enrutado de los distintos paquetes de estimación y/o *feedback* se realiza salto a salto, implicando a todos los nodos de la red.

- **Routing entre *cluster-heads***: Aquí, aunque en las tareas de sensado, computación y en la comunicación de las medidas obtenidas en cada T_s intervienen todos los nodos que conforman la red, el enrutado no se realiza salto a salto, sino entre los *cluster-heads* de la red. Los nodos son vecinos si se encuentran dentro de un radio de transmisión R, y, si, o bien pertenecen al mismo *cluster* que dicho nodo, o bien son *cluster-heads* de *clusters* vecinos.

5.8. Fusión de paquetes de estimación

Para cumplir uno de los objetivos planteados al comienzo de este estudio, en esta sección se plantea un sistema de fusión de datos aplicable a los algoritmos de filtros de partículas distribuidos planteados en las secciones anteriores.

Se pueden utilizar técnicas de fusión de datos en los paquetes de estimación que viajan a través de la red de nodos hacia el *sink* aprovechando las características de la función de verosimilitud global explicadas en la ecuación (5.21) de la Sección 5.1, pues esta se construye a partir de las funciones de verosimilitud locales para un mismo instante k a través de la expresión:

$$p\left(\mathbf{z}_k|\mathbf{x}_k^{(m)}\right) = \prod_{c=1}^{C} p\left(\mathbf{z}_k^c|\mathbf{x}_k^{(m)}\right) \tag{5.29}$$

Inicialmente cada paquete de estimación es enviado desde el *cluster-head* que lo construye con una función de verosimilitud local determinada. Si un nodo i de la red recibe un paquete de estimación construido a partir de observaciones en el instante k por el cluster c_1, $\mathbf{z}_k^{c_1}$, y en su memoria hay al menos otro paquete de estimación construido en el mismo instante a partir de las observaciones de otro grupo de nodos distinto al primero,$\mathbf{z}_k^{c_2}$, se puede construir un nuevo paquete que sustituya a ambos y contenga la información de ambas verosimilitudes locales $\left\{\mathbf{z}_k^{c_1},\ \ \mathbf{z}_k^{c_2}\right\}$

$$p\left(\mathbf{z}_k^{c_1,c_2}|\mathbf{x}_k^{(m)}\right) = p\left(\mathbf{z}_k^{c_1}|\mathbf{x}_k^{(m)}\right) \cdot p\left(\mathbf{z}_k^{c_2}|\mathbf{x}_k^{(m)}\right) \tag{5.30}$$

donde:

- $p\left(\mathbf{z}_k^{c_1}|\mathbf{x}_k^{(m)}\right)$ es la función de verosimilitud local en el instante k calculada en el *cluster* c_1
- $p\left(\mathbf{z}_k^{c_2}|\mathbf{x}_k^{(m)}\right)$ es la función de verosimilitud local en el instante k calculada en el *cluster* c_2
- $p\left(\mathbf{z}_k^{c_1,c_2}|\mathbf{x}_k^{(m)}\right)$ es la función de verosimilitud en el instante k asociada a los *clusters* c_1 y c_2, resultado de la fusión de datos.

El proceso de fusión se puede observar en la Figura 5.7. Para posibilitar la construcción correcta de la verosimilitud global asociada a cada instante k en el nodo destino, en los paquetes de estimación se deberá añadir información sobre el número de verosimilitudes locales que se encuentran agregadas, a través de un campo nuevo dentro del paquete.

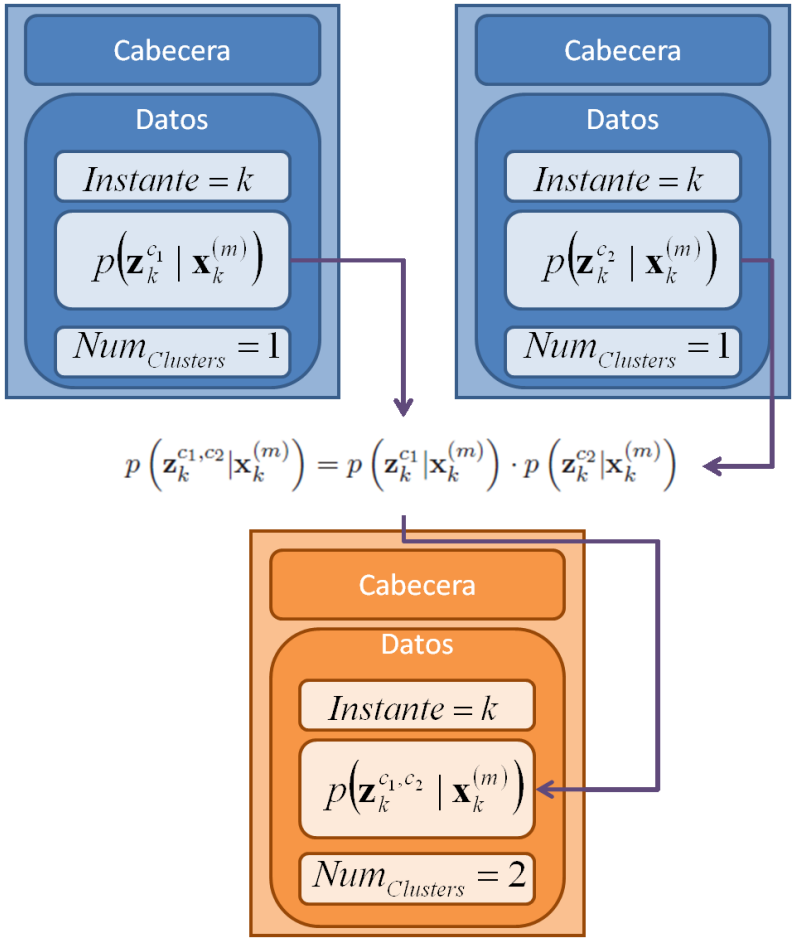


Figura 5.7: Fusión de datos de dos paquetes de estimación generados para el mismo instante k

Capítulo 6

SIMULACIONES

En este capítulo se presentan los resultados de las simulaciones realizadas con cada uno de los esquemas propuestos en el capítulo anterior.

6.1. Parámetros de simulación

Para comprobar el funcionamiento de los esquemas definidos en el *Capítulo 5*, analizar los resultados y comparar su eficiencia, se han realizado simulaciones utilizando Matlab®.

El área en la que se despliega la red de sensores es un cuadrado de $40x40m^2$, con 64 sensores desplegados en una estructura de rejilla, separados aproximadamente $5m$ entre sí.

Los nodos se agrupan en *clusters* de 4 nodos, habiendo $C = 16$ *clusters* dentro de la red inicialmente. Cada cluster posee un líder o *cluster-head* que al comienzo de la simulación es el nodo situado en la esquina derecha-superior de cada *cluster*. El destino o nodo *sink* se encuentra desplegado en la esquina superior derecha de la red. La estructura física de la red desplegada en las simulaciones que se han realizado en este estudio puede observarse en la Figura 6.1.

Los nodos se encuentran en funcionamiento realizando las tareas de sensado, computación, comunicación y enrutado correspondientes hasta que se quedan sin batería. Si un nodo distinto al *sink* detecta que su batería es mínima y va a morir, envía esta información a sus vecinos. En caso de que sea un *cluster-head*, pasa el testigo al primer nodo vecino de su *cluster*.

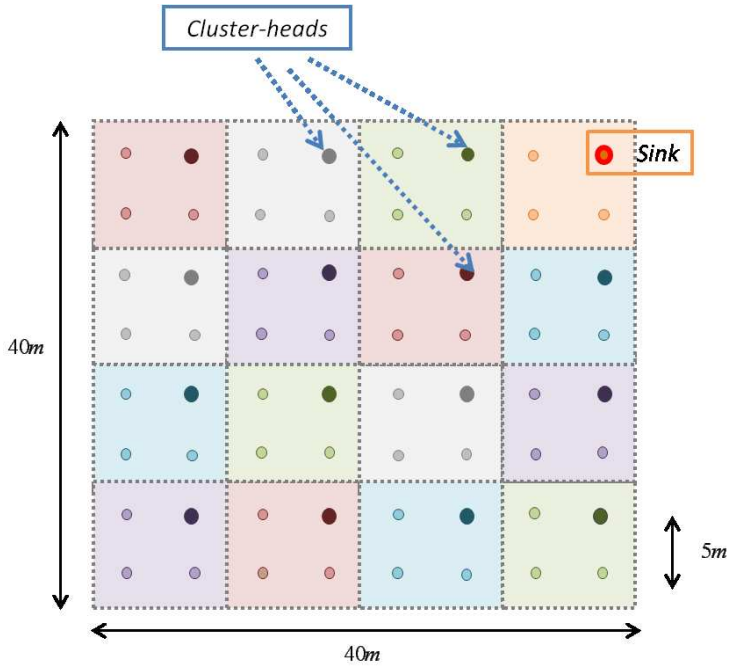


Figura 6.1: Estructura física de la red desplegada en las simulaciones

Todos los nodos poseen una batería inicial $E = 50000$ unidades de energía (u.e.) exceptuando el nodo *sink*, cuyo suministro de energía es ilimitado. Se utiliza un modelo de energía determinístico caracterizado por tres parámetros constantes y conocidos:

- E_R, energía gastada en la recepción de mensajes.
- E_T, energía gastada en la transmisión de mensajes.
- E_I, energía gastada en el sensado y obtención de observaciones.

donde, $E_R = 2$, $E_T = 5$,$E_I = 2$ u.e.

Todos los nodos tienen antenas omnidireccionales y áreas recíprocas de cobertura. Se asume que los nodos son conscientes de las transmisiones realizadas en su área de cobertura y utilizan esta información para realizar la estimación de los parámetros. Cada nodo sabe su propia localización, la de sus vecinos - incluyendo el *cluster-head* asociado a su *cluster*, y la localización del nodo *sink* o destino. Los nodos realizan tareas de enrutado cada $T_r = 0{,}1s$, y tareas de sensado cada T_s, valor que será modificable en las simulaciones.

Teniendo en cuenta el modelo de medición descrito en la Sección 3.2 que define el modelo de sensor, se han utilizado los siguientes valores para los distintos parámetros en la ecuación 3.11

- La potencia recibida a la distancia de referencia, $P_0 = 0dBm$.
- La distancia de referencia, $d_0 = 1m$.
- El exponenente de pérdidas en el camino se ha fijado en $\alpha = 2$
- La varianza del ruido $\sigma_n = 1dB$, con media nula.

Aunque existen numerosas estrategias de enrutado de paquetes, en las simulaciones se ha utilizado *Greedy Forwarding* por simplicidad. Así, si un nodo ha de enviar un mensaje hacia el destinatario, lo enviará a aquel vecino cuya distancia con el nodo *sink* sea menor.

El nodo *sink* tiene un radio de alcance mucho mayor que ningún otro nodo de la red, y es capaz de enviar en difusión mensajes de *feedback* a todos los nodos que la conforman. Las pérdidas por enlace no se han tenido en cuenta en los modelos desarrollados.

Tal y como se comentó en el *Capítulo 5*, se ejecutará un filtro de partículas por cada *cluster*. Tendremos por tanto $C = 16$ filtros de partículas ejecutándose de forma sincronizada, según el esquema general del filtro de partículas distribuido desarrollado en la Sección 5.1. El número de partículas existente en cada uno de ellos es $M = 500$.

Las trayectorias del blanco han sido generadas según el modelo de movimiento presente en la Sección 3.1, lineal y dinámico en el tiempo, descrito por la ecuación 3.6. Las matrices $\mathbf{G}_x$ y $\mathbf{G}_u$ dependen del valor del periodo de sensado, T_s, que varía según las simulaciones.

El blanco siempre se encuentra dentro del campo de acción de la red y se mueve con una velocidad variable entre $[0.v_{max}]$, siendo $v_{max} = 2m/s$.

Para cada uno de los modelos determinados en el *Capítulo 5* se aplican los dos tipos de algoritmos de *routing* descritos en la Sección 5.7 del mismo capítulo. El routing completo, en el que todos los nodos que forman la red tienen sus capacidades de *routing* activadas. Y el routing entre

cluster-heads donde los nodos que no actúan como *cluster-heads* no participan en el enrutado, solamente en caso de que un *cluster-head* les pase el testigo de líder del *cluster* correspondiente.

En el Modelo 2 el nodo *sink* decide que el remuestreo es necesario si una cuarta parte de los *cluster-heads* lo necesitan, es decir, si 4 *cluster-heads* han activado el flag de remuestreo en los paquetes de estimación asociados a un instante concreto.

En aquellos modelos en los que se utilice el concepto de *número de partículas eficaces*, N_{eff}, esto es, en los Modelos 2 y 3, existen dos subvariantes de funcionamiento dependiendo si el umbral N_{th} utilizado para comparar N_{eff} (y determinar así si el remuestreo es o no necesario en una iteración completa) es fijo o variable con el tiempo ($N_{th}(k)$). En este segundo caso y según lo dispuesto en la ecuación 5.26, los parámetros de la función que define dicho umbral han sido fijados a los siguientes valores:

- $N_{th_{max}} = M = 500$
- $N_{th_{min}} = \frac{2 \cdot M}{3} = 333.\widehat{3}$, límite recomendado en la Sección 4.2.1
- $A = -\frac{25}{2}$ y $B = \frac{5}{4}$

En las variantes con umbral fijo, N_{th}, se ha dispuesto su valor al límite recomendado, $N_{th} = \frac{2 \cdot M}{3}$.

En los Modelos 2 y 3 se ha decidido que el tiempo máximo que un nodo debe esperar dependa del número de nodos inactivos en la red, según lo desarrollado en la Sección 5.6. Esta función se define como $T_{wait}(k)$ a través de la siguiente ecuación 5.28, donde se han fijado sus parámetros a los valores siguientes:

- $T_{wait_{max}} = 15$ y $T_{wait_{min}} = 4$ para routing entre *cluster-heads*,
- $T_{wait_{max}} = 20$ y $T_{wait_{min}} = 8$ con routing completo.
- $A = -\frac{25}{8}$ y $B = \frac{5}{32}$

Se distinguen tres tipos de mensajes que circulan a través de la red:

- ***Mensajes de sensado*** $\Longrightarrow$ Aquellos que contienen información sobre la observaciones tomadas por un nodo en un instante k. Son generados por cada uno de los nodos activos de la red en cada periodo T_s y enviados desde estos al nodo que funciona como *cluster-head* en cada uno de los *clusters* correspondientes.

- ***Mensajes de estimación*** $\Longrightarrow$ Aquellos generados por cada *cluster-head* en cada periodo T_s, con el cómputo local del filtro de partículas correspondiente usando la información obtenida con los mensajes de sensado recibidos. Se dirigen desde cada *cluster-head* hacia el nodo *sink* y siendo objeto de técnicas de fusión de datos en su camino hacia el destino.

- ***Mensajes de retroalimentación o feedback*** $\Longrightarrow$ Aquellos generados por el nodo *sink*, en caso de que, según el algoritmo de filtro de partículas lo determine, sea necesario un remuestreo de las partículas a través de la red. Son mensajes enviados en difusión a todos los nodos de la red, conteniendo los índices de aquellas partículas que han de ser remuestreadas y el número de réplicas correspondientes.

Se utilizan políticas de fusión de datos en todos los nodos de la red para disminuir el número de mensajes que circulan a través de ella. Así, si un nodo recibe un mensaje de estimación asociado a un instante k_i y tiene otro mensaje de estimación asociado al mismo instante dentro de su buffer de envío, aplica técnicas de fusión de datos (Sección 5.8) para crear un único mensaje que enviar al siguiente salto.

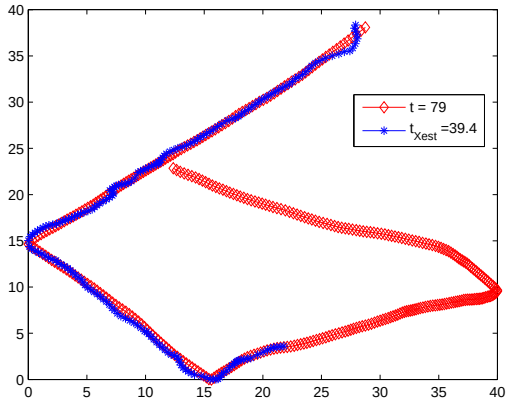
La evaluación de cada uno de los modelos propuestos en el *Capítulo 5* se ha realizado en términos de error de estimación del blanco, RMSE, vida útil de la red, *TTL - Time To Live*, retardo de las estimaciones obtenidas y número de iteraciones del algoritmo por cada remuestreo, con el fin de determinar las ventajas e inconvenientes de cada uno de los algoritmos. Se calcula el TTL como la diferencia existente desde que comienza la simulación hasta que el nodo *sink* se queda aislado, momento en el que se considera que la red muere. El retardo se calcula como el promedio de la diferencia de tiempo existente entre el instante en el que calcula cada una de las estimaciones y el instante k en el que fueron obtenidas las observaciones para esa estimación.

Para obtener resultados más generales, las simulaciones se han promediado sobre 100 realizaciones para cada uno de los modelos y sus variantes y en las que se ejecutan distintas trayectorias.

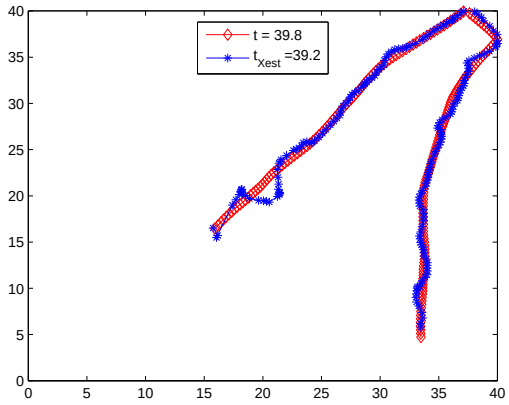
El tiempo de simulación máxima predeterminado es $t = 6000s$. Los resultados analizados se presentan en la siguiente sección.

6.2. Resultado de las simulaciones

El funcionamiento de la aplicación de seguimiento queda ilustrado en las Figuras 6.2(a) y 6.2(b). En ambas la trayectoria del blanco se presenta en color rojo, y en azul las estimaciones calculadas. Se puede ver la convergencia del sistema tras una serie de iteraciones. Así, el error asociado a las estimaciones obtenidas en las primeras iteraciones es mayor que en las siguientes. Además, se observa el retardo presente entre la última estimación calculada y la posición del blanco en el instante en el que ha sido calculada.



(a) Estimación de la trayectoria con retardo



(b) Estimación de la trayectoria sin retardo

Figura 6.2: Ejemplos de estimación de la trayectoria del blanco

6.2.1. Modelo 1

En esta versión del Esquema General propuesto, descrita en la Sección 5.2 todos los *cluster-heads* de la red poseen una misma constante, irm, que indica el número de iteraciones que pueden ejecutar el filtro de partículas antes de realizar el paso de remuestreo. En ese instante esperarán un mensaje de *feedback* enviado por el nodo destino con los índices de remuestreo asociados a las partículas del filtro. Los valores fijados para irm en las simulaciones son $irms = \{0 \;\; 1 \;\; 2 \;\; 3 \;\; 4\}$. Los valores elegidos para el periodo de observación son $T_s = \{0{,}1 \;\; 0{,}2 \;\; 0{,}3\}\,[s]$.

La Tabla 6.1 muestra los resultados obtenidos en las simulaciones realizadas cuando el enrutado se lleva a cabo únicamente entre *cluster-heads*, mientras que en la Tabla 6.2 se observan aquellos correspondientes al uso de un routing completo, en el que participan todos los nodos de la red. Se recuerda que el Esquema General coincide con el Modelo 1 cuando $irms = 0$.

irm	T_s **[s]**	**TTL [s]**	**Retardo [s]**	**Remuestreos**	**Estimaciones**
0	0.1	934.5	373.911	1848	1848
	0.2	1567.8	471.0591	3121	3121
	0.3	2021.8	406.5797	4024	4024
1	0.1	943.7	314.9543	3060	6120
	0.2	1571.7	264.0506	5201	10402
	0.3	2132.2	6.0144	6543	13086
2	0.1	949.7	266.2199	3917	11751
	0.2	1643.1	122.0117	6500	19500
	0.3	2251.7	0.30975	6830	20490
3	0.1	954.1	239.5797	4556	18224
	0.2	1659	5.3756	7539	30156
	0.3	2314.3	0.29797	8275	33100
4	0.1	957.1	213.9602	5046	25230
	0.2	1681.6	0.3731	7941	39705
	0.3	2354	0.29054	8640	43200

Tabla 6.1: Resultado de las simulaciones en el Modelo 1 para routing entre *cluster-heads*

En la Tabla 6.1 podemos observar el tiempo de vida útil de la red frente al periodo de sensado, T_s, para cada una de las variantes del Modelo 1 cuando el routing se realiza entre los líderes de cada *cluster*. Los resultados muestran que el TTL asociado aumenta con T_s para un mismo irm. Este crecimiento está también presente con el aumento de $irms$ para un T_s fijo, aunque las diferencias con T_s cercanos al periodo de enrutado no resultan significativas y van aumentando con T_s mayores.

El retardo asociado decrece con T_s para irm fijos, excepto en los casos $irm = 0$ e $irm = 1$ en los que se presentan valores mayores con $T_s > T_r$. La misma observación puede hacerse con el retardo presente para un mismo T_s variando el valor de irm. Este decrece con el aumento de $irms$, con ciertas excepciones en $T_s = 0{,}1s$ o $T_s = 0{,}2s$. Ya que el retardo depende del tiempo de vida de la red, observando únicamente los valores obtenidos en las simulaciones no podemos sacar conclusiones. Más adelante se muestra esta dependencia en la Figura 6.4.

Se puede ver que a mayor irm o T_s mayor número de estimaciones. Resultado que resulta lógico, pues el número de estimaciones de la posición del blanco obtenidas depende directamente del tiempo en el que la red se encuentre activa, y, se ha observado con anterioridad que el TTL asociado para el Modelo 1 aumenta con ambos parámetros, $irms$ y T_s.

El número de estimaciones obtenidas es directamente proporcional al número de remuestreos. Depende del parámetro $irms$, siendo el número medio de estimaciones calculadas equivalente a $irm + 1$ veces el número medio de remuestreos realizados.

En la Tabla 6.2 se presentan los mismos resultados que en la tabla anterior pero para el caso en el que routing se realice entre todos los nodos de la red. Los resultados son similares al caso anterior, en el que el enrutado se realizaba entre *cluster-heads*:

- El tiempo de vida de la red aumenta tanto con T_s como con irm.
- El retardo asociado decrece con T_S e irm excepto en algunos casos.
- Se observa que a mayor irm o T_s mayor número de estimaciones.
- El número medio de estimaciones calculadas equivalente a $irm + 1$ veces el número medio

irm	T_s [s]	TTL [s]	Retardo [s]	Remuestreos	Estimaciones
0	0.1	791.7	323.6716	833	833
	0.2	1317.8	437.8617	1316	1316
	0.3	1569.7	467.7101	1857	1857
1	0.1	890.7	346.5071	955	1910
	0.2	1422.3	389.1996	2577	5154
	0.3	1630.3	270.4565	2843	5686
2	0.1	869.9	300.9776	1380	4140
	0.2	1492.7	295.8191	3123	9369
	0.3	1684.9	96.5201	3343	10029
3	0.1	879.5	278.681	1853	7412
	0.2	1520.5	206.3688	3648	14592
	0.3	1790.1	1.6609	3896	15584
4	0.1	882.4	233.8728	2631	13155
	0.2	1667.5	150.5061	4054	20270
	0.3	1836.1	1.572	4219	21095

Tabla 6.2: Resultado de las simulaciones en el Modelo 1 para routing completo

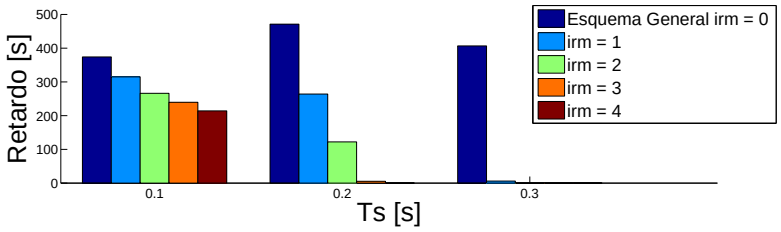
de remuestreos realizados.

Si se compara el tiempo de vida de la red para unos irm y T_s determinados en las dos tablas, la red permanece activa un periodo de tiempo mayor en caso de que el routing se produzca entre *cluster-heads*, alrededor de $1{,}2 - 1{,}3$ veces mayor. Esta diferencia es lógica puesto que el número de saltos necesario para que un paquete llegue a destino es menor cuando el routing se realiza entre *cluster-heads*. Diferencia que se ve incrementada al aumentar T_s puesto que el tiempo entre observaciones es mayor y, por tanto también el tiempo entre generación de paquetes.

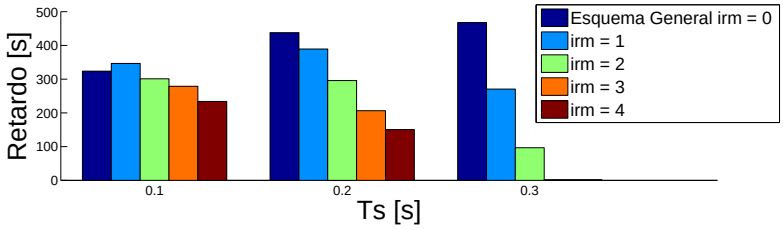
Comparando ambas tablas, las simulaciones con routing completo presentan un mayor retardo que aquellas con el primer algoritmo de routing para valores altos de irm y T_s. Todo lo contrario ocurre para valores menores. Aunque esta diferencia no parece lógica, debido a que el retardo depende en gran medida del tiempo en el que la red haya estado activa, mostraremos

más adelante la dependencia de ambos valores, TTL y retardo en la estimación del blanco.

El número de estimaciones obtenidas para una configuración concreta de la red es, en el caso de utilizar un enrutado entre *cluster-heads* entre $2-2{,}3$ veces mayor que cuando todos los nodos de la red se encuentran implicados.



(a) Modelo 1: Retardo promedio para routing entre *cluster-heads*



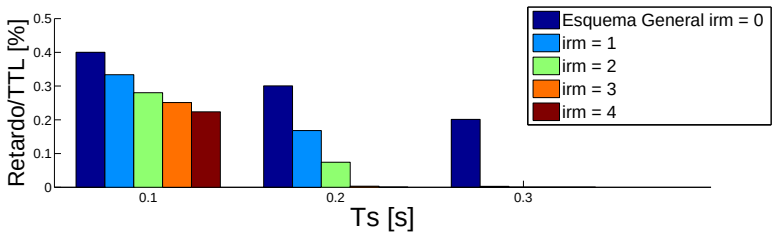
(b) Modelo 1: Retardo promedio para routing completo

Figura 6.3: Modelo 1: Retardo promedio para $irms = \{0 \;\; 1 \;\; 2 \;\; 3 \;\; 4\}$

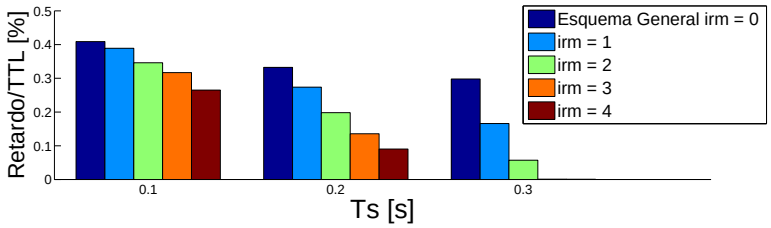
En la Figura 6.3 se puede observar el retardo asociado a cada una de las variantes del Modelo 1, esto es con $irms = \{0 \;\; 1 \;\; 2 \;\; 3 \;\; 4\}$ en función del periodo de sensado T_s. En este caso se puede ver como, al contrario de lo que podría esperarse el retardo existente en la estimación de la posición del blanco no decrece en todos los casos al aumentar el periodo de muestreo. Para aquellas variantes en las que $irm > 0$ esta disminución sí se da cuando el routing es entre *cluster-head*, y en caso de $irm > 1$ cuando el routing es completo. Los resultados muestran que el retardo es inexistente para $T_s \geq 3 \cdot T_r$, excepto para el caso general en el que se puede considerar que la estimación se realiza a tiempo real a partir de $T_s > 3 \cdot T_r$.

Además, cuando el routing se realiza entre *cluster-heads*, el retardo asociado a cada uno de los modelos es ligeramente menor que si se realiza entre todos los nodos que conforman la red. Como este retardo depende del tiempo de vida de la red, aumentando cuando esta crece, para

tener una visión más objetiva de las diferencias se ha calculado la relación entre el retardo y la vida de la red $\frac{retardo}{TTL}$, cuyos resultados pueden observarse en la Figura 6.4.



(a) Modelo 1: Retardo/TTL promedio para routing entre *cluster-heads*



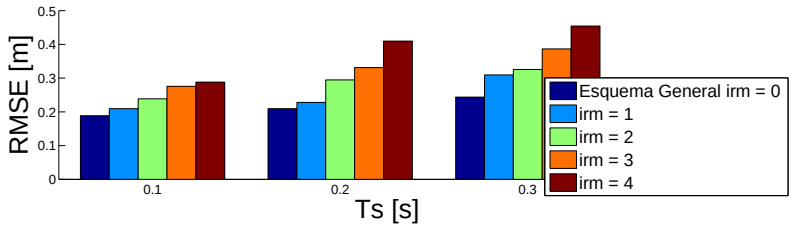
(b) Modelo 1: Retardo/TTL promedio para routing completo

Figura 6.4: Modelo 1: Retardo/TTL promedio para $irms = \{0 \;\; 1 \;\; 2 \;\; 3 \;\; 4\}$

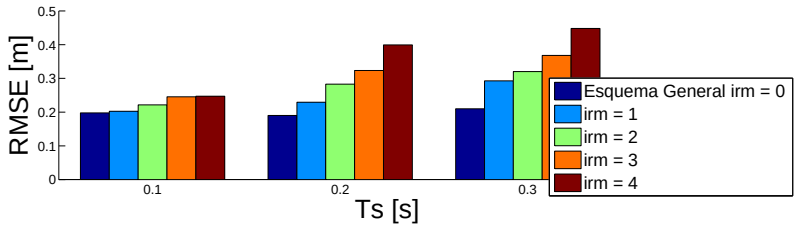
En esta figura se muestra el ratio del retardo frente al TTL obtenido para cada una de las configuraciones del Modelo 1. El peor caso corresponde al esquema general. En este la estimación del blanco tiene un retardo del 0,4 % del tiempo de vida de la red para un $T_s = T_r$; relación que disminuye con el periodo de sensado hasta valores cercanos al 0,25 %. En general el ratio $\frac{retardo}{TTL}$ disminuye tanto con T_s como con $irms$. En el mejor caso, cuando la estimación se puede considerar a tiempo real, el promedio es cercano al 0 %.

El RMSE obtenido en las versiones del Modelo 1 se muestra en la Figura 6.5. En él se puede observar que el menor error corresponde en cada caso con el modelo correspondiente al esquema general, pues realiza remuestreo en todas las iteraciones del algoritmo. Por otro lado, el error aumenta con el periodo de sensado, T_s. Además, a mayor irm mayor RMSE; diferencias que en el caso en el que el periodo de routing y el de sensado coincidan $T_s \equiv T_r = 0{,}1s$ resultan mínimas, del orden de $0{,}02m$, y que van aumentando con mayores valores de T_s.

En la última figura que se muestra para el primer modelo (Figura 6.6) se ha calculado el

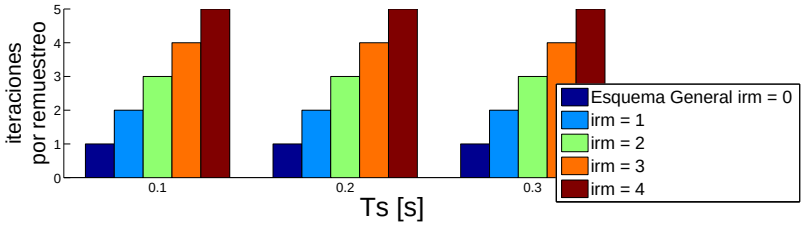


(a) Modelo 1: RMSE promedio para routing entre *cluster-heads*

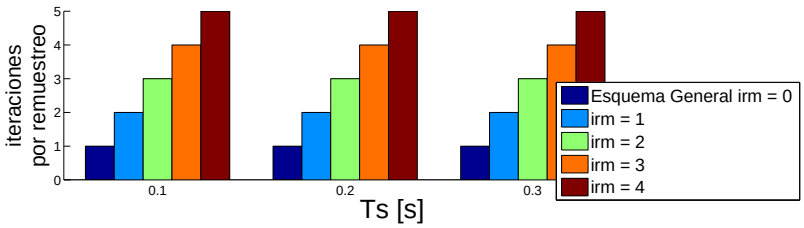


(b) Modelo 1: RMSE promedio para routing completo

Figura 6.5: Modelo 1: RMSE promedio para $irms = \{0 \;\; 1 \;\; 2 \;\; 3 \;\; 4\}$



(a) Modelo 1: iteraciones/remuestreos para routing entre *cluster-heads*



(b) Modelo 1: iteraciones/remuestreos promedio para routing completo

Figura 6.6: Modelo 1: iteraciones/remuestreos promedio para $irms = \{0 \;\; 1 \;\; 2 \;\; 3 \;\; 4\}$

número medio de iteraciones del algoritmo por cada remuestreo. Se puede observar que en este caso es idéntica independientemente del periodo de sensado elegido y que corresponde a $irm + 1$. Así por ejemplo, en el Esquema General siempre remuestreará, esto es, realiza 1 iteración para un remuestreo, mientras que en el segundo caso, en el que $irm = 1$ realizará 2 iteraciones por cada remuestreo, es decir, una iteración sin remuestrear y a la siguiente remuestrea. Estas medidas

son independientes del algoritmo de enrutado, por lo que coinciden para las simulaciones con routing completo y routing entre *cluster-heads*.

6.2.2. Modelo 2

En este modelo todos los *cluster-heads* utilizan el concepto de partículas eficaces, N_{eff}, definido en la Sección 4.2.1. Cada uno de los *cluster-heads* calcula en cada iteración del algoritmo del filtro de partículas el número de partículas eficaces correspondiente N_{eff}^{c}, con $c = 1 \cdots C$ siendo C el número de *clusters* de la red.

En cada paquete de estimación, cada *cluster-head* activará o no un flag específico dependiendo de su necesidad de remuestreo. Es el nodo *sink* el que determina, una vez ha recibido toda la información, si el remuestreo resulta necesario (si el número de *cluster-heads* que necesitan remuestreo supera un umbral, que en las simulaciones ha sido fijado a 4. Los valores elegidos para el periodo de muestreo son los mismos que para el modelo anterior, $T_s = \{0{,}1 \;\; 0{,}2 \;\; 0{,}3\}\,[s]$.

Además, existen dos variantes de este modelo, dependiendo del umbral N_{th} con el que se compara N_{eff} en cada caso. En el primer caso el umbral es fijo en el tiempo. En el segundo varía, $N_{th}(k)$, siendo mayor en los primeros instantes para intentar reducir el tiempo de convergencia del algoritmo.

En las Tablas 6.3 y 6.4 se pueden observar los resultados medios obtenidos en las simulaciones realizadas. En el primer caso aquellas cuyo algoritmo de enrutado es entre *cluster-heads*; el segundo corresponde a un routing completo. Dichos resultados se muestran también en las gráficas que se incluyen en esta sección.

Observando el tiempo de vida útil de la red frente al periodo de observación, T_s, el TTL asociado aumenta con T_s puesto que el tiempo entre observaciones (y entre generación de paquetes) es mayor. Los resultados obtenidos son similares para los dos tipos de umbral N_{th}.

Se puede considerar que el retardo es también similar en los dos tipos de umbral utilizados, fijo y variable con el tiempo. Aunque se observa que el retardo es creciente con T_s se compara

N_{th}	T_s [s]	TTL [s]	Retardo [s]	Remuestreos	Estimaciones
fijo	0.1	1057.5	314.4038	280.12	1788
	0.2	1930.254	421.5532	888.25	3254.18
	0.3	2635.176	361.211	1432.32	4449.45
variable	0.1	1057.3	314.9282	299.57	1773
	0.2	1928.873	421.5213	904.38	3253.53
	0.3	2636.712	359.5004	1434.17	4434.58

Tabla 6.3: Resultado de las simulaciones en el Modelo 2 para routing entre *cluster-heads*

más adelante la relación existente entre estos dos parámetros, el tiempo de vida de la red y el retardo producido, pues existe una dependencia entre ellos.

El número de estimaciones obtenidas es también similar para umbrales fijos o variables del número de partículas eficaces. Este número de estimaciones crece con el periodo de observación, T_s. Además, a mayor T_s mayor número de remuestreos. Resulta importante saber el número medio de iteraciones del algoritmo necesarias para realizar un único remuestreo. Esta relación se muestra más adelante y permite comparar el funcionamiento del Modelo 2 con el Esquema General.

N_{th}	T_s [s]	TTL [s]	Retardo [s]	Remuestreos	Estimaciones
fijo	0.1	955.6	330.4731	133.41	966
	0.2	1508.378	418.7587	360.87	1546.96
	0.3	1754.889	445.3911	609.2	2021.15
variable	0.1	952.3	334.6917	144.15	931
	0.2	1505.668	414.1228	374.3	1566.15
	0.3	1751.592	430.939	623.35	2044.97

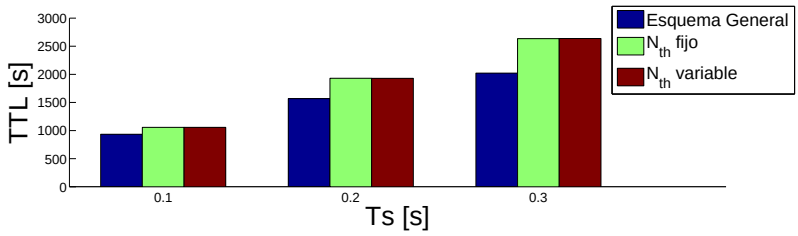
Tabla 6.4: Resultado de las simulaciones en el Modelo 2 para routing completo

Comparando la Tabla 6.4 con la tabla anterior se puede observar que el comportamiento es

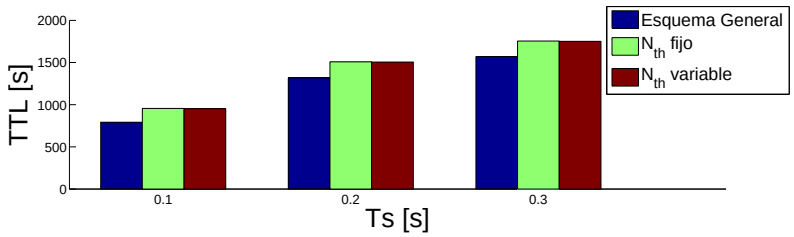
equivalente en el mismo caso, y las observaciones arriba realizadas para routing entre *cluster-heads* se pueden interpolar aquí:

- Los resultados son similares para ambos tipos de umbral N_{th}: fijo y variable.
- El TTL asociado aumenta con T_s.
- El retardo es creciente con el aumento de T_s.
- El número de estimaciones aumenta con T_s.
- El número de remuestreos crece con el tiempo de observación T_s.

Se observan diferencias entre los valores de cada uno de los parámetros asociados a una misma configuración del Modelo 2, pero con distintos enrutados. En el caso del tiempo de vida de la red, este es $1{,}1 - 1{,}5$ veces mayor para routing entre líderes del cluster. El retardo, por otro lado, resulta cerca de 1,2 veces mayor si el routing es completo. En las simulaciones con routing entre *cluster-heads* se obtienen cerca del doble de estimaciones del blanco. Lo mismo ocurre con el número medio de remuestreos.



(a) Modelo 2: TTL promedio para routing entre *cluster-heads*

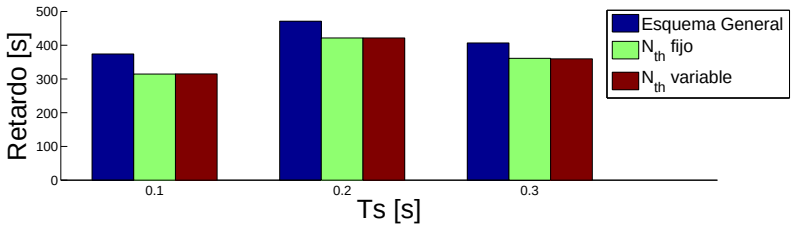


(b) Modelo 2: TTL promedio para routing completo

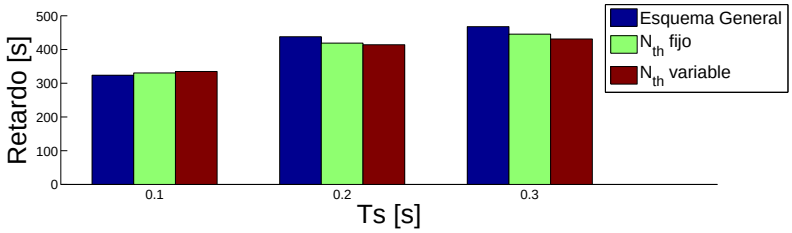
Figura 6.7: Modelo 2: TTL promedio para N_{th} fijo y variable

En la Figura 6.7 puede observarse el tiempo de vida útil de la red frente al periodo de sensado, T_s, para cada una de las variantes del Modelo 2. En color verde los resultados correspondientes

a un umbral fijo del número de partículas eficaces, N_{th}, y en rojo el obtenido con un umbral variable $N_{th}(k)$. Se muestra además el valor correspondiente al Esquema General, que aparecerá como referencia en cada una de las gráficas presentes en esta sección. Puede observarse que en ambas variantes del Modelo 2 el tiempo de vida obtenido es superior al correspondiente al Esquema General. Esta diferencia aumenta con el periodo de sensado, ya que, además la vida de la red aumenta en general con este parámetro.



(a) Modelo 2: Retardo promedio para routing entre *cluster-heads*

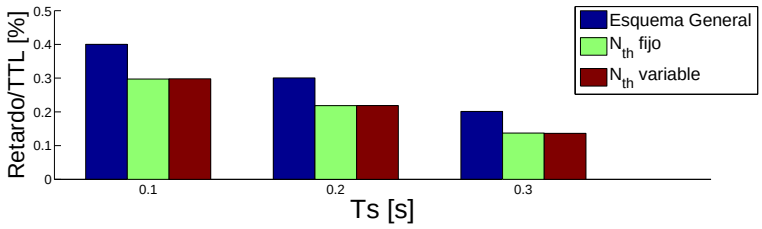


(b) Modelo 2: Retardo promedio para routing completo

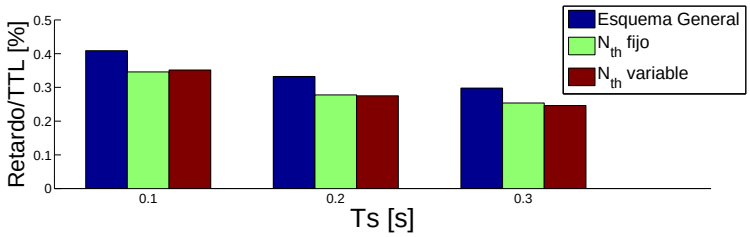
Figura 6.8: Modelo 2: Retardo promedio para N_{th} fijo y variable

En la Figura 6.8 se muestra el retardo asociado a las simulaciones del Modelo 2 frente al obtenido con el Esquema General. En todos los casos el retardo obtenido es menor que el calculado para el Esquema General, siendo los retardos del routing completo menores que para el routing entre *cluster-heads*, tal y como ocurría en el modelo anterior. Además, se observa que el retardo es creciente con T_s. Como este retardo depende del tiempo de vida de la red, aumentando cuando esta crece, para tener una visión más objetiva se muestra la relación entre el retardo y la vida de la red $\frac{retardo}{TTL}$ en la siguiente figura.

En la Figura 6.9 se observa el ratio entre el retardo calculado frente al tiempo que la red permanece activa. Este ratio es continuamente decreciente con T_s y menor que para el Esquema General propuesto. Los valores obtenidos entre los distintos algoritmos de enrutado son superio-



(a) Modelo 2: Retardo/TTL promedio para routing entre *cluster-heads*



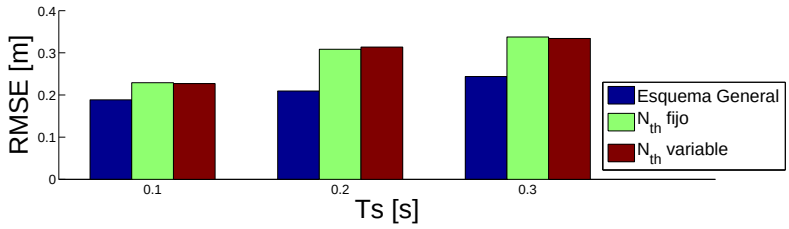
(b) Modelo 2: Retardo/TTL promedio para routing completo

Figura 6.9: Modelo 2: Retardo/TTL promedio para N_{th} fijo y variable

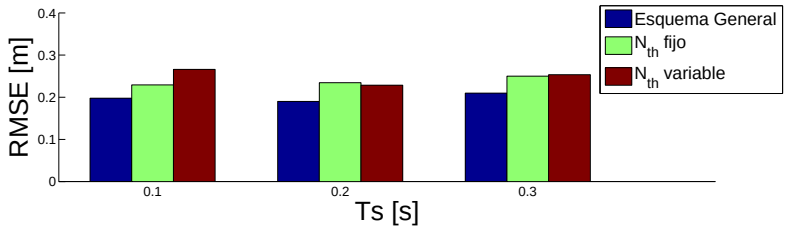
res en el routing completo. Así, para $T_s \equiv T_r = 0{,}1s$ el retardo en la estimación corresponde a un 0,3 % de la vida de la red si el enrutado se lleva a cabo entre *cluster-heads*, mientras que para un routing completo el retardo corresponde a un 0,35 % del TTL asociado, para el mismo caso. La diferencia aumenta con el periodo de sensado, siendo para $T_s = 0{,}3s$ del 0,15 % con el primer routing y del 0,3 % para el routing completo.

En la Figura 6.10 podemos observar el error RMSE obtenido para cada una de las versiones del Modelo 2 así como los valores correspondientes al esquema general. El error cuadrático medio calculado es siempre mayor que en el esquema general, aumentando su diferencia conforme aumenta el valor del periodo de muestreo. Comparando los resultados obtenidos entre las versiones del Modelo 2, umbral fijo y variable del número de partículas eficaces, podemos apreciar diferencias entre ambos valores. Estas, del orden de $\{0{,}02 - 0{,}04\}\,[m]$, no resultan significativas.

En la última figura correspondiente a este modelo observamos las iteraciones realizadas del filtro de partículas por cada remuestreo. En el esquema general este valor es siempre igual a 1, pues realiza un remuestreo en cada iteración. En el Modelo 2 depende del número de *cluster-heads* que necesiten remuestrear en cada uno de los instantes. Los valores resultan similares tanto para las dos variantes del modelo, esto es, en caso de usar umbrales fijos o variables en el número

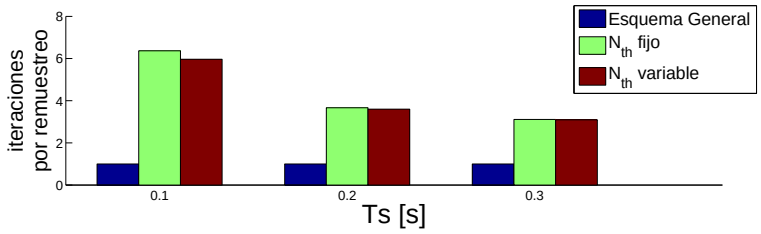


(a) Modelo 2: RMSE promedio para routing entre *cluster-heads*

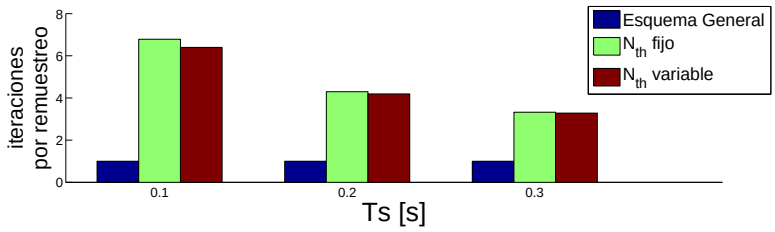


(b) Modelo 2: RMSE promedio para routing completo

Figura 6.10: Modelo 2: RMSE promedio para N_{th} fijo y variable



(a) Modelo 2: iteraciones/remuestreos para routing entre *cluster-heads*



(b) Modelo 2: iteraciones/remuestreos promedio para routing completo

Figura 6.11: Modelo 2: iteraciones/remuestreos promedio para N_{th} fijo y variable

de partículas eficaces, como para los distintos algoritmos de routing. Para valores de $T_s \equiv T_r$ se realizan entre 6 y 7 iteraciones por remuestreo. Este número decrece con el aumento de T_s, tendiendo a 3 iteraciones por remuestreo para $T_s \geq 0{,}3s$.

6.2.3. Modelo 3

El funcionamiento del Modelo 3 es similar al del modelo anterior, introduciendo una modificación en la forma de calcular la necesidad de remuestreo. Todos los *cluster-heads* de la red ejecutan el filtro de partículas de forma distribuida, pero, a diferencia del Modelo 2 no calculan el número de partículas eficaces N_{eff} en cada instante sino que envían únicamente la función de verosimilitud local que han calculado al destino. Una vez realizada esta parte del algoritmo se colocan en modo espera. El nodo *sink*, una vez recibida la información de cada uno de los nodos, calcula la función de verosimilitud completa y el número de partículas eficaces N_{eff}, comparando este con un umbral determinado para decidir si se remuestrea o no.

Del mismo modo que en el modelo anterior, en el Modelo 3 existen dos variantes dependiendo del umbral N_{th} con el que se compara N_{eff} en cada caso. En el primer caso el umbral es fijo en el tiempo. En el segundo varía, $N_{th}(k)$, siendo mayor en los primeros instantes para intentar reducir el tiempo de convergencia del algoritmo.

En las Tablas 6.5 y 6.6 se pueden observar los resultados medios obtenidos en las simulaciones realizadas. El primer caso corresponde a aquellas simulaciones configuradas con un algoritmo de routing entre *cluster-heads*, mientras que en la segunda tabla se muestran los resultados correspondientes a un routing entre todos los nodos de la red.

N_{th}	$T_s[s]$	**TTL [s]**	**Retardo [s]**	**Remuestreos**	**Estimaciones**
fijo	0.1	1215.4	333.3802	1166.813	2280.45
	0.2	1939.477	414.8526	2852.8165	4225.652
	0.3	2382.625	270.6671	4764.8755	5718.1105
variable	0.1	1215.4	333.9805	1183.902	2298.85
	0.2	1939.477	415.4507	2865.616	4229.4815
	0.3	2382.625	269.8582	4765.692	5706.047

Tabla 6.5: Resultado de las simulaciones en el Modelo 3 para routing entre *cluster-heads*

En la primera tabla se muestra que TTL, tiempo de vida útil de la red, asociado aumenta con

T_s. Esta tendencia se observaba también en los dos modelos anteriores ya que el tiempo entre observaciones aumenta y con él el tiempo entre generación de paquetes. Los valores de TTL obtenidos son similares para ambos tipos de umbral N_{th}. Se puede considerar que los valores del retardo son también similares en ambas subvariantes.

El número de estimaciones obtenidas es también similar para umbrales fijos o variables del número de partículas eficaces. Este número de estimaciones crece con el periodo de observación, T_s. Además, a mayor T_s mayor número de remuestreos.

N_{th}	$T_s[s]$	**TTL [s]**	**Retardo [s]**	**Remuestreos**	**Estimaciones**
fijo	0.1	1128.5	328.611	535.348	1136.2
	0.2	1555.089	436.3549	1204.28	1781.8215
	0.3	1764.715	427.0469	1673.917	2206.1715
variable	0.1	1128.5	333.0849	557.5545	1108.6
	0.2	1555.089	432.2781	1196.6325	1767.8835
	0.3	1764.715	433.8059	1710.142	2240.3725

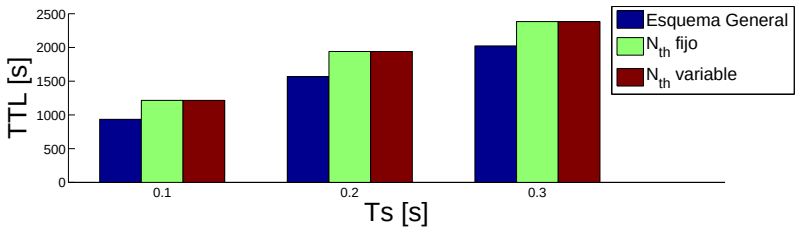
Tabla 6.6: Resultado de las simulaciones en el Modelo 3 para routing completo

Comparando ambas tablas se puede concluir que el comportamiento de los parámetros frente a las variaciones de T_s es similar en ambos tipos de enrutado:

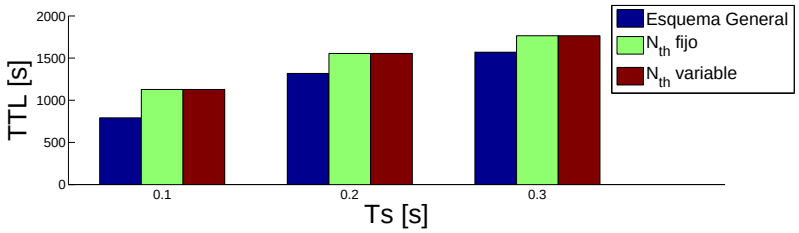
- Los resultados son similares para ambos tipos de umbral N_{th}: fijo y variable.
- El TTL asociado aumenta con T_s
- El número de estimaciones aumenta con T_s.
- El número de remuestreos crece con el tiempo de observación T_s.

Aún así, aunque las tendencias de los parámetros sean similares, se observan diferencias entre los valores asociados a una misma configuración, pero con distintos enrutados. En el caso del tiempo de vida de la red, este es $1{,}1 - 1{,}35$ veces mayor para routing entre líderes del *cluster*. El retardo, por otro lado, resulta cerca de $1{,}2 - 1{,}8$ veces mayor si el routing es completo, siendo

creciente con T_s en el caso de utilizar un enrutado entre todos los nodos de la red. Debido a estos resultados, será necesario ver el ratio de retardo en función del TTL. Estos resultados se presentan más adelante. En las simulaciones con routing entre *cluster-heads* se obtienen cerca del doble de estimaciones del blanco. Lo mismo ocurre con el número medio de remuestreos.



(a) Modelo 3: TTL promedio para routing entre *cluster-heads*

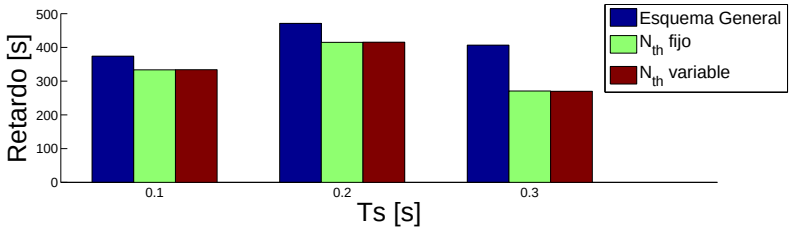


(b) Modelo 3: TTL promedio para routing completo

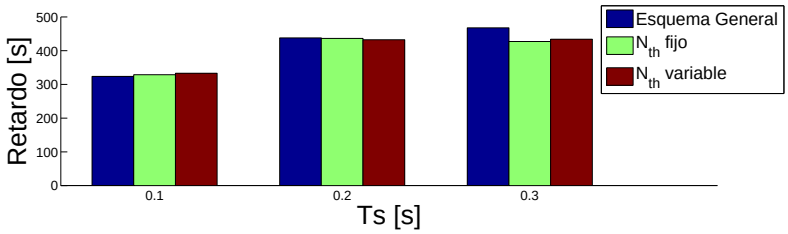
Figura 6.12: Modelo 3: TTL promedio para $irms = \{0 \;\; 1 \;\; 2 \;\; 3 \;\; 4\}$

En la Figura 6.12 se muestra el tiempo de vida útil de la red frente al periodo de sensado, T_s, para cada una de sus variantes. En color verde los resultados correspondientes a un umbral fijo del número de partículas eficaces, N_{th} y en rojo el obtenido con un umbral variable N_{th}. Se muestra además el valor correspondiente al Esquema General propuesto, que aparecerá como referencia en cada una de las figuras presentes en esta sección. Puede observarse que, al igual que ocurría en el anterior modelo, en ambas variantes tiempo de vida obtenido es similar, siendo superior al correspondiente al Esquema General. Esta diferencia aumenta con el periodo de sensado, ya que, además la vida de la red aumenta en general con este periodo.

De forma similar a los modelos anteriores, el algoritmo de routing influye en el tiempo de vida de la red. La red permanece activa menor tiempo en caso de utilizar el algoritmo completo tal y como se había comentado anteriormente al comparar los resultados de las Tablas 6.5 y 6.6.



(a) Modelo 3: Retardo promedio para routing entre *cluster-heads*



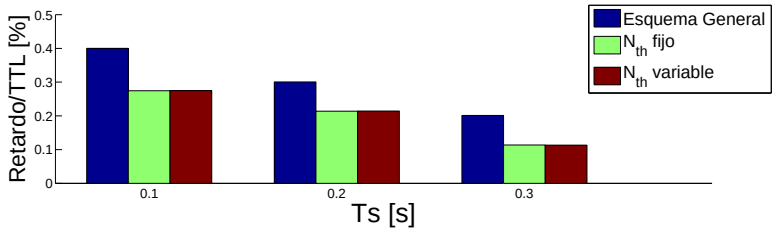
(b) Modelo 3: Retardo promedio para routing completo

Figura 6.13: Modelo 3: Retardo promedio para N_{th} fijo y variable

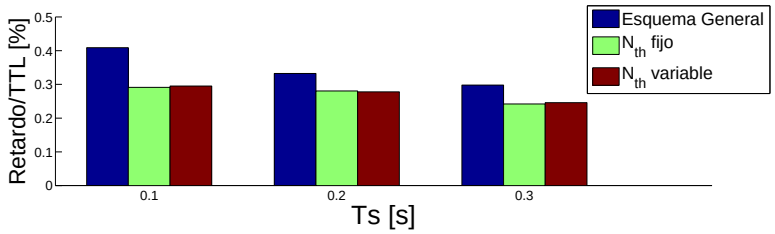
El retardo asociado a las simulaciones del Modelo 3 frente al obtenido con el Esquema General se puede observar en la Figura 6.13. Para routing entre líderes del *cluster* el retardo es menor que en el caso del Esquema General, mientras que los retardos del routing completo son similares a este, y ligeramente menores en el caso de $T_s = 0{,}3s$. El tipo de umbral, fijo o variable, del número de partículas eficaces no afecta a este parámetro ya que proporciona resultados similares.

En la Figura 6.14 se muestra el retardo calculado frente al tiempo que la red permanece activa. Se puede ver que este ratio es continuamente decreciente con T_s, por lo que el retardo depende del tiempo de vida en el que la red permanece activa. La diferencia por tanto observada en la anterior figura vendría determinada por el aumento del tiempo de vida entre ambos valores de periodo de sensado. Existe una diferencia considerable entre el ratio de cualquiera de las variantes del Modelo 3 y el obtenido para el Esquema General. Diferencia, por ejemplo, del 0,13 % para $T_s \equiv T_r$ para routing entre *cluster-heads*; y, ligeramente inferior en el caso de un algoritmo de enrutado completo, del 0,11 %.

En la Figura 6.15 podemos observar el error RMSE obtenido para cada una de las versiones del Modelo 3 así como los valores correspondientes al Esquema General. Los valores de RMSE aumentan con el periodo de observación T_s. El error es similar al obtenido con el Esquema Ge-

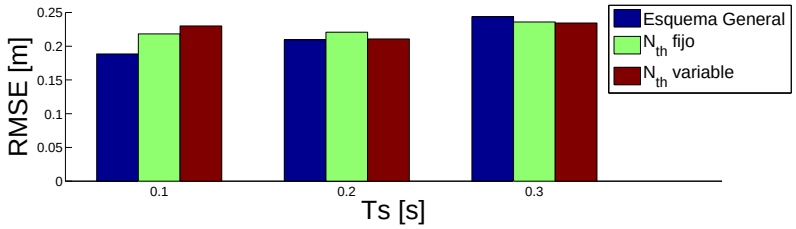


(a) Modelo 3: Retardo/TTL promedio para routing entre *cluster-heads*

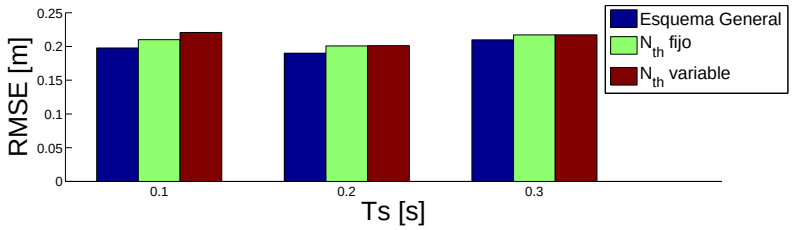


(b) Modelo 3: Retardo/TTL promedio para routing completo

Figura 6.14: Modelo 3: Retardo/TTL promedio para N_{th} fijo y variable



(a) Modelo 3: RMSE promedio para routing entre *cluster-heads*

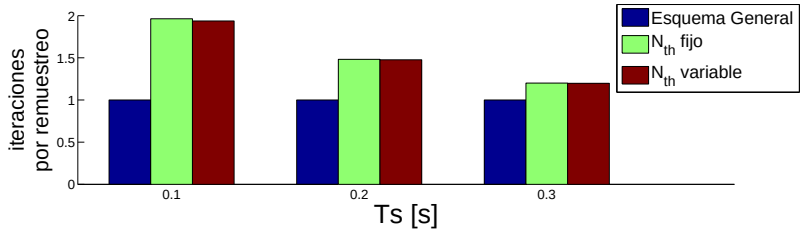


(b) Modelo 3: RMSE promedio para routing completo

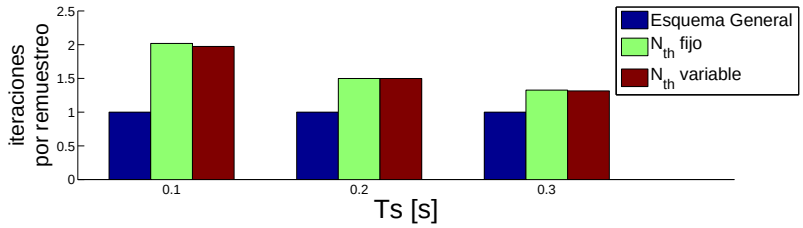
Figura 6.15: Modelo 3: RMSE promedio para N_{th} fijo y variable

neral y similar para ambos tipos de umbral de partículas eficaces, pues las diferencias existentes son del orden de $0{,}02 - 0{,}03m$ y no resultan significativas.

Resulta importante saber el número medio de iteraciones del algoritmo necesarias para realizar un único remuestreo pues esta relación permite comparar el funcionamiento del Modelo 2 con



(a) Modelo 3: iteraciones/remuestreos para routing entre *cluster-heads*



(b) Modelo 3: iteraciones/remuestreos promedio para routing completo

Figura 6.16: Modelo 3: iteraciones/remuestreos promedio para N_{th} fijo y variable

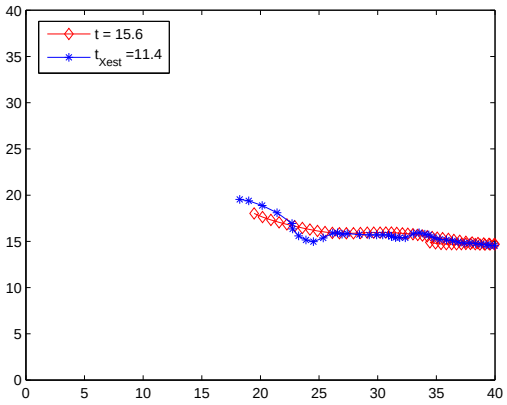
el Esquema General. En la última figura correspondiente a este modelo observamos las iteraciones realizadas del filtro de partículas por cada remuestreo. En el Esquema General este valor es siempre igual a 1, pues realiza un remuestreo en cada iteración. En el Modelo 3 depende del número de partículas eficaces calculadas en cada iteración por el *sink*, que decidirá comparando este valor con el umbral establecido en cada variante si el remuestreo es o no necesario. Los valores resultan similares tanto para las dos variantes del modelo, esto es, en caso de usar umbrales fijos o variables en el número de partículas eficaces, como para los distintos algoritmos de routing. Para valores de $T_s \equiv T_r$ se realizan aproximadamente 2 iteraciones por cada remuestreo. Esta cifra disminuye al aumentar el valor de T_s siendo en $T_s = 0{,}3s$ de 1,2 iteraciones por remuestreo y, tendiendo hacia 1 iteración por remuestreo con valores mayores de T_s.

6.2.4. Umbral fijo/variable para el número de partículas eficaces

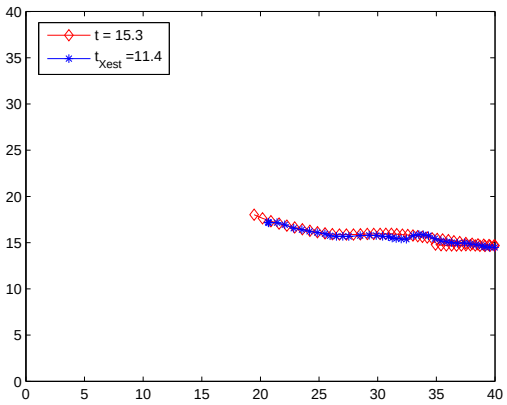
En las secciones anteriores correspondientes a los Modelos 2 y 3 se contaba con dos subvariantes de funcionamiento, dependiendo de las características del umbral utilizado para comparar el número de partículas eficaces calculadas en cada iteración y determinar la necesidad de remuestreo según la degeneración del grupo de partículas. La primera utilizaba un umbral fijo en el tiempo; en la segunda variante el umbral era variable en cada instante k.

Ya que los valores de los parámetros utilizados para calibrar el funcionamiento de los modelos son similares en ambos casos (RMSE, TTL, retardo, etc.), se hace necesario realizar estudio distinto al anterior para observar la diferencia entre ambos casos.

En un modelo con $N_{th}(k)$ el valor del umbral en los primeros instantes es máximo, y equivale al número de partículas M. Conforme avanza el número de instantes k, este umbral se va modificando hasta alcanzar el valor recomendado de $2M/3$. Este último valor es el utilizado en modelos con umbral constante. Ya que la variación entre M y $2M/3$ se realiza en los primeros $k_i = 20s$ de la simulación.



(a) Modelo 2: convergencia del filtro con umbral fijo



(b) Modelo 2: convergencia del filtro con umbral variable

Figura 6.17: Modelo 2: Ejemplos de convergencia del filtro de partículas

La ventaja de utilizar un umbral variable en la comparación de partículas eficaces frente a

uno fijo radica en el tiempo de convergencia inicial del algoritmo. Con un umbral variable, el tiempo de convergencia es menor.

Este efecto puede observarse en las Figuras 6.17. Las dos imágenes corresponden a los primeros instantes del algoritmo, donde se ha aplicado la misma configuración (en este caso corresponde al Modelo 2, routing entre *cluster-heads*, $T_s = 0{,}2s$) e idéntica trayectoria para las simulaciones. En la Figura 6.17(a), con umbral fijo, la convergencia del algoritmo es mayor, de aproximadamente 12 iteraciones. Utilizando un umbral variable (Figura 6.17(b)), el tiempo de convergencia es prácticamente nulo, pues en los primeros instantes se obliga al sistema a remuestrear siempre, con lo que las partículas no sufren efecto de degeneración.

Aunque se podría esperar un número de remuestreos mayor en el caso de tener un umbral variable, y que estos afectaran a otros parámetros como el error cometido en las simulaciones, el efecto de variar el umbral no se ve reflejado en los parámetros obtenidos en las simulaciones pues el TTL es mucho más elevado que el tiempo en el que el umbral varía inicialmente.

La elección de uno u otro dependerá de la características y necesidades propias del entorno donde se vaya a aplicar el sistema desarrollado.

Capítulo 7

CONCLUSIONES

Dentro de este capítulo se presentan una serie de conclusiones de las distintas soluciones propuestas en el *Capítulo 5*. Conclusiones basadas en los resultados de las simulaciones, expuestos en el *Capítulo 6*. Se compararán todos los modelos con el Esquema General descrito en 5.1, proponiéndose un modelo único, atendiendo a las ventajas e inconvenientes de su implementación. Al final del capítulo se proponen algunas posibles pinceladas de cómo continuar desarrollando este estudio en un futuro.

7.1. Conclusiones del estudio

Las comparaciones entre los distintos modelos que se establecen en esta sección son generales y no tienen en cuenta las diferencias existentes entre las simulaciones con distinto algoritmo de routing. Las conclusiones referentes al enrutado se presentan de forma independiente y común a todas las soluciones.

El Modelo 1 presenta mejores prestaciones que el Esquema General en términos de TTL, número medio de estimaciones calculadas y mejores ratios de retardo frente al tiempo de vida de la red. En esta primera modificación del Esquema General se puede considerar que, para el tamaño de red elegido, existe retardo nulo utilizando $irms \uparrow$. En términos de RMSE, nos encontramos con peores prestaciones del Modelo 1 frente al Esquema General. Así, a mayor irm mayor error cometido, pues se aumenta el número de iteraciones del filtro de partículas en las que los líde-

res de cada cluster no realizan remuestreo y, por tanto, la desincronización de las partículas de cada líder aumenta y con ello se produce mayor efecto de degeneración de las partículas del filtro.

La desventaja principal del Modelo 1 radica en la necesidad de una alta sincronización de los contadores irm_c de cada *cluster* entre sí y con el contador irm_S del nodo destino. Una desincronización de un *cluster-head* puede suponer que este espere siempre un mensaje de feedback asociado a un instante de observación que nunca le llegará. El nodo destino se vería forzado a esperar en cada iteración el tiempo límite determinado, y realizaría las estimaciones sin poseer la información completa de la red.

Como conclusión podemos establecer que, aunque el mejor modelo dependerá de las características del entorno en el que se vaya a aplicar, el Modelo 1 logra el objetivo de mejorar la vida útil de la red respecto al Esquema General propuesto, disminuyendo también el retardo que aparece entre la obtención de la estimación del blanco y la posición de este. Sin embargo, como la aplicación de este modelo supone una degeneración de la estimación, no se recomienda su aplicación para valores altos de irm, ya que el error cometido podría ser demasiado grande.

Los problemas de sincronización del Modelo 1 hicieron que se propusiera una nueva variación del Esquema General, en la que la necesidad de sincronización no fuera tan fuerte. El Modelo 2 elimina la necesidad de contadores, pero introduce una computación añadida en cada uno de los filtros de partículas que corren en cada *cluster-head*. En cada iteración los líderes de cada *cluster* deben calcular el número de partículas eficaces y decidir si necesitan o no realizar remuestreo. La decisión final de este paso queda en manos del nodo destino.

El Modelo 2, al igual que el Modelo 1 presenta mejores prestaciones que el Esquema General en términos de TTL, número medio de estimaciones calculadas y ratio de retardo frente al tiempo de vida de la red. Sin embargo, a diferencia del Modelo 1 el número de estimaciones obtenidas es menor y los retardos calculados mayores, no alcanzando en ningún momento retardo 0. Desde el punto de vista de RMSE las prestaciones del Modelo 2 son más bajas que en el Esquema General y similares al Modelo 1 en el caso en el que $irms = 2$.

Para el Modelo 3 hemos comprobado que las prestaciones en términos de TTL son mejores

que en cualquier otra solución propuesta. Los valores de retardo del Modelo 3 son menores que los determinados por el Esquema General y también menores que los asociados al Modelo 2, aunque mayores a los asociados al Modelo 1, no alcanzándose retardo nulo para las características de la red simulada. El número de estimaciones obtenidas tampoco llega a superar a las del Modelo 1, por la necesidad de esperar de los *cluster-heads* antes de tomar la decisión de remuestreo; sin embargo las estimaciones son mayores que las presentes en el Modelo 2 y el Esquema General. En términos de error, el Modelo 3 mejora con respecto al funcionamiento del Modelo 2 y Modelo 1, obteniéndose valores similares a los pertenecientes al Esquema general.

Como conclusión podemos establecer que, aunque el mejor modelo dependerá de las características del entorno en el que se vaya a aplicar, se recomienda la utilización del Modelo 3. Esta solución logra el objetivo de mejorar la vida útil de la red respecto al Esquema General propuesto, disminuyendo también el ratio del retardo con respecto a la vida útil de la red. La ventaja existente con respecto al Modelo 1 es que no necesitan tener una sincronización muy alta en los tiempos de observación. Respecto al Modelo 2, el Modelo 3 mejora los ratios de error en las estimaciones y elimina la computación del número de partículas eficaces en los nodos líderes de cada cluster, pues este cálculo y la decisión de remuestreo cae del lado del nodo *sink*.

Por otro lado, la comparación de ambos algoritmos de enrutado nos puede llevar a hacernos una idea de cómo escalan estos esquemas propuestos. Aunque el error cuadrático medio o el número de iteraciones por remuestreo es independiente de la escala de la red, el tiempo de vida de la red y el retardo asociado sí dependen de esta escala. A mayor número de saltos, más energía se consume en el trayecto de un paquete a través de la red hacia el nodo destino. Por ello el tiempo necesario para que el nodo destino tenga la información completa para estimar la posición en un instante determinado será mayor y con ello disminuirá el número de estimaciones realizadas.

En general, para un tamaño de red fijo, se recomienda la utilización del routing entre *cluster-heads*. Se ha de tener en cuenta una limitación para la aplicación de este algoritmo. La primera es la distancia entre nodos y, entre *cluster-heads*. La energía necesaria para transmitir un paquete aumenta exponencialmente con la distancia de transmisión, aunque aquí se haya establecido un gasto fijo para cada envío de paquetes. Por tanto, si la distancia entre nodos es muy grande, se recomienda el routing salto a salto, entre todos los nodos de la red, tal y como se explica en la

ecuación 2.2.

En cuanto a las dos subvariantes presentes en los Modelos 2 y 3 referentes al umbral con el que se compara el número de partículas eficaces, se han observado prestaciones similares en cada uno de las soluciones. La diferencia principal entre ambos modelos es el número de iteraciones necesarias para la convergencia inicial del algoritmo, siendo mucho menor en el caso de utilizar un umbral variable en el tiempo. Aunque la elección de una u otra subvariante dependerá de las necesidades propias del entorno donde se vaya a desarrollar la aplicación, en general no se recomienda el uso de un umbral variable, a no ser que la velocidad de convergencia inicial resulte esencial para la aplicación, por la mayor complicación en la programación de la implementación y la mayor necesidad de computación del algoritmo en cada iteración.

7.2. Líneas Futuras de estudio y desarrollo

Se pretende desarrollar el siguiente trabajo futuro, relacionado con el proyecto:

- Implementar los modelos propuestos en otros entornos de programación distintos a MATLAB® para mejorar la eficiencia del algoritmo, permitiendo la ejecución en paralelo de cada uno de los filtros de partículas que forman el sistema.
- Generalizar el sistema para poder estimar la posición de múltiples blancos en movimiento dentro del área de la red.
- Reducir el tamaño de los paquetes de estimación, buscando formas más compactas para caracterizar las funciones de verosimilitud, que permitan seguir aplicando métodos de fusión de datos en el camino.

APÉNDICES

APÉNDICE A

PLANIFICACIÓN Y PRESUPUESTO DEL PROYECTO

En este apéndice se describen las tareas que se han llevado a cabo durante la ejecución de este Proyecto de Fin de Carrera. Se incluye además un desglose justificado del presupuesto total del proyecto, detallando el coste tanto material como de personal asociado a cada una de las tareas ejecutadas. Tales costes se pueden deducir de la Tabla A.1 y el desglose de la dedicación de cada una de las fases se puede observar en la Tabla A.2.

A.1. Descripción de las tareas

En esta sección se encuentran listadas y detalladas las tareas que se han realizado durante la ejecución de este proyecto, describiendo su finalidad, la relación con otras tareas, la duración, el esfuerzo dedicado a cada una de ellas y el coste asociado.

A.1.1. Tarea A: Documentación

Subtarea A1: *Revisión y estudio de la documentación asociada a las nociones de redes de sensores inalámbricos*

- *Descripción:* Estudio detallado de las redes de sensores inalámbricos, sus características y sus aplicaciones.

- *Objetivos:* Profundizar en los conceptos de las redes de sensores inalámbricos.
- *Dependencia con otras tareas:* Es la tarea que supone el inicio del proyecto.
- *Duración:* 1 semana.
- *Recursos:* Ingeniero Superior (0,5 hombres/mes).

Subtarea A2: *Revisión del estado del arte de las redes de sensores en la resolución del problema de detección y seguimientos de blancos en movimiento*

- *Descripción:* Esta tarea busca la familiarización de los conceptos y técnicas asociadas en aplicaciones de detección y seguimiento de blancos en redes de sensores.
- *Objetivos:* Adquirir las nociones básicas de las técnicas típicamente utilizadas en aplicaciones de este tipo. Visualizar el funcionamiento de un escenario típico.
- *Dependencia con otras tareas:* Esta tarea da comienzo tras finalizar la subtarea A1.
- *Duración:* 1 semana.
- *Recursos:* Ingeniero Superior (0,5 hombres/mes).

Subtarea A3: *Revisión del estado del arte de los Algoritmos de Filtros de Partículas*

- *Descripción:* En esta fase de la actividad de documentación se consolidan los conocimientos necesarios para la implementación de un algoritmo de filtro de partículas. Se estudian sus características y propiedades así como las distintas formas de implementación dentro de una red de sensores inalámbricos.
- *Objetivos:* Entender el funcionamiento de los filtros de partículas y su aplicación en aplicaciones de detección y seguimientos de blancos, profundizando en su base teórica.
- *Dependencia con otras tareas:* Esta tarea da comienzo tras finalizar la subtarea A2. Su finalización posibilitará el inicio de la siguiente subtarea.
- *Duración:* 1 semana.

- *Recursos:* Ingeniero Superior (0,5 hombres/mes).

Subtarea A4: *Revisión del estado del arte de los Algoritmos de Filtros de Partículas Distribuidos*

- *Descripción:* En esta última fase de la actividad de documentación se investiga la aplicación de filtros de partículas en sistemas distribuidos, profundizando en los principales esquemas que posibilitan la distribución de la carga computacional del algoritmo.
- *Objetivos:* Adquirir los conocimientos necesarios sobre la aplicación de algoritmos de filtros de partículas en sistemas distribuidos.
- *Dependencia con otras tareas:* Esta tarea da comienzo tras finalizar la subtarea A3.
- *Duración:* 3 semanas.
- *Recursos:* Ingeniero Superior (0,5 hombres/mes).

A.1.2. Tarea B: Desarrollo del Software

Subtarea B1: *Desarrollo de funciones que simulen una red de sensores*

- *Descripción:* Desarrollo de las funciones que simulen:
  - la generación de una red de sensores situados físicamente a través de una configuración de rejilla;
  - la determinación de los vecinos de cada uno de los nodos;
  - la división del conjunto de nodos simulados en *clusters*.
- *Objetivos:* Programar el código necesario que simule una red de N sensores desplegados en un área A y que se encuentren divididos en C *clusters*. Donde N, A y C sean configurables.
- *Dependencia con otras tareas:* Es la tarea que supone el inicio del desarrollo del software y permitirá, tras su finalización, el comienzo de la tarea siguiente. Se inicia tras la finalización de las tareas de documentación.
- *Duración:* 3 semanas.

- *Recursos:* Ingeniero Superior (0,25 hombres/mes).

Subtarea B2: *Desarrollo de los algoritmos de routing*

- *Descripción:* Desarrollo de las funciones que implementen los dos algoritmos de routing: «salto a salto» y «entre cluster-heads» definidos previamente.
- *Objetivos:* Programar el código necesario que permita el paso de mensajes vacíos desde cualquier nodo de la red al nodo destino y viceversa según las características de los algoritmos de enrutado definidos previamente. Introducir el gasto de batería al generar paquetes, transmitirlos o recibirlos.
- *Dependencia con otras tareas:* Se inicia una vez finalizada la subtarea B1 y posibilitará el comienzo de la tarea siguiente.
- *Duración:* 3 semanas.
- *Recursos:* Ingeniero Superior (0,25 hombres/mes).

Subtarea B3: *Desarrollo de las funciones de reconfiguración de la red tras la muerte de sensores*

- *Descripción:* Desarrollar las funciones necesarias que permitan una reconfiguración de la red tras la muerte de un sensor, hasta que el nodo destino se quede aislado.
- *Objetivos:* Programar el código necesario para:
  - actualizar la configuración de los vecinos en el caso de que un nodo se vaya a quedar sin batería;
  - posibilitar el paso del testigo de líder del *cluster* en caso de que el nodo sea un *cluster-head*.
- *Dependencia con otras tareas:* Se inicia una vez finalizada la subtarea B2 y posibilitará el comienzo de la tarea siguiente.
- *Duración:* 2 semanas.

- *Recursos:* Ingeniero Superior (0,25 hombres/mes).

Subtarea B4: *Implementación del algoritmo genérico de Filtros de Partículas de forma distribuida*

- *Descripción:* En esta tarea se programa el algoritmo genérico del filtro de partículas así como todas las funciones necesarias para su ejecución.
- *Objetivos:* Programar las funciones necesarias que permitan el funcionamiento de un sistema de partículas distribuidas según el esquema general propuesto y la estimación de la posición del blanco
- *Dependencia con otras tareas:* Se inicia una vez finalizada la subtarea B3 y posibilitará el comienzo de la tarea siguiente.
- *Duración:* 3 semanas.
- *Recursos:* Ingeniero Superior (0,5 hombres/mes).

Subtarea B5: *Implementación de las variantes al algoritmo genérico de filtro de partículas distribuido*

- *Descripción:* Programar las variantes del algoritmo genérico del filtro de partículas distribuidas según las decisiones de diseño tomadas.
- *Objetivos:* Programar las funciones necesarias que permitan implementar las distintas variantes del esquema distribuido de filtro de partículas
- *Dependencia con otras tareas:* Se inicia una vez finalizada la subtarea B4 y posibilitará el comienzo de la tarea siguiente.
- *Duración:* 8 semanas.
- *Recursos:* Ingeniero Superior (0,25 hombres/mes).

Subtarea B6: *Ajuste del funcionamiento las distintas versiones del algoritmo genérico*

- *Descripción:* Ajuste de los parámetros de inicialización que caracterizan cada uno de los modelos de filtros de partículas implementados.
- *Objetivos:* Ajustar y determinar los parámetros de inicialización que caracterizarán a cada versión del esquema general propuesto.
- *Dependencia con otras tareas:* Se inicia una vez finalizada la subtarea B5 y posibilitará el comienzo de la tarea siguiente.
- *Duración:* 6 semanas.
- *Recursos:* Ingeniero Superior (0,25 hombres/mes).

A.1.3. Tarea C: Obtención de simulaciones

Subtarea C1: *Realización de las pruebas preliminares*

- *Descripción:* Programar una batería de pruebas del software que permitan comprobar su correcto comportamiento y eliminar posibles errores.
- *Objetivos:* Comprobar el correcto funcionamiento del software implementado y elegir las simulaciones a realizar.
- *Dependencia con otras tareas:* Se inicia una vez finalizada la subtarea B6 y posibilitará el comienzo de la tarea siguiente.
- *Duración:* 2 semanas.
- *Recursos:* Ingeniero Superior (0,25 hombres/mes).

Subtarea C2: *Simulaciones definitivas y análisis de los resultados*

- *Descripción:* Ejecución de las simulaciones definitivas para cada uno de los modelos de filtros de partículas propuestos, utilizando los parámetros de ajuste calculados en la subtarea B6. Realización de un primer análisis de los resultados obtenidos y repetición de las simulaciones en caso de ser necesario.

- *Objetivos:* Ejecutar las simulaciones propuestas y realizar un análisis de los resultados obtenidos en estas.
- *Dependencia con otras tareas:* Se puede comenzar a ejecutar una semana después de que comience la subtarea B1, cuando se dispongan de versiones del software sin errores.
- *Duración:* 9 semanas.
- *Recursos:* Ingeniero Superior (0,5 hombres/mes).

A.1.4. Tarea D: Redacción de la memoria

Subtarea D1: *Organización y estructura de la memoria del documento*

- *Descripción:* Definición y organización de la estructura de la memoria del proyecto
- *Objetivos:* Determinar los capítulos y secciones y estructurar la memoria de este Proyecto de Fin de Carrera.
- *Dependencia con otras tareas:* Esta tarea puede dar comienzo una vez que se tengan definidas todas las versiones del algoritmo, unas semanas después del comienzo de la subtarea B5.
- *Duración:* 1 semana.
- *Recursos:* Ingeniero Superior (0,25 hombres/mes).

Subtarea D2: *Realización del documento final*

- *Descripción:* Redacción del documento final del proyecto.
- *Objetivos:* Redactar y desarrollar cada uno de los capítulos de este Proyecto de Fin de Carrera, así como de sus apéndices.
- *Dependencia con otras tareas:* Una primera parte de esta subtarea comienza una vez terminada la fase de documentación (Subtarea A.3). El resto comenzará una vez que se obtengan los primeros resultados preliminares y se compruebe el funcionamiento correcto de las simulaciones definitivas (4 semanas después de la subtarea C1)

- *Duración:* 13 semanas.
- *Recursos:* Ingeniero Superior (0,50 hombres/mes).

A.1.5. Tarea E: Presentación del proyecto

- *Descripción:* Preparación de la presentación de este Proyecto de Fin de Carrera
- *Objetivos:* Elaborar un conjunto de diapositivas adecuado que permitan tener una visión clara y completa del trabajo realizado
- *Dependencia con otras tareas:* Se inicia una vez finalizada la subtarea E2.
- *Duración:* 1 semanas.
- *Recursos:* Ingeniero Superior (0,75 hombres/mes).

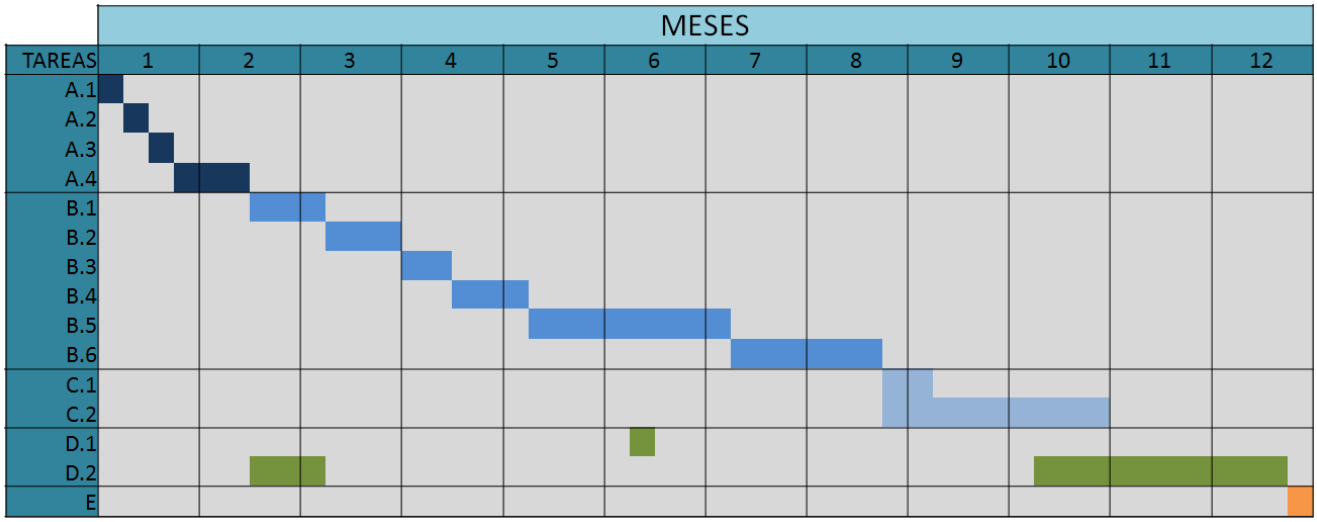


Figura A.1: Diagrama de Gantt de las fases del proyecto

En la Figura A.1 se muestra el diagrama de Gantt de la ejecución de las tareas de este proyecto. En esta gráfica se puede apreciar de forma sencilla la distribución temporal de cada tarea dentro de la duración global del proyecto.

A.2. Recursos

En esta sección detallamos los recursos empleados en la realización de este Proyecto Fin de Carrera.

Los recursos materiales utilizados han sido:

- Documentación -*IEEE Member Digital Library (monthly subscription)*-.
- Ordenador portátil con procesador Intel Core 2 Duo T7300 2GHz. Sistema operativo Windows XP.
- Programa MATLAB®, versión R2010.

La Tabla A.1 recoge los costes de material y los costes humanos desglosados según los recursos arriba listados.

Concepto	**Cantidad**	**Coste [€]**	**Total [€]**
Ordenador	1	1.300€	1.300€
Documentación	12	30€	360€
Licencia MATLAB®	1	500€	500€
1 Ingeniero Superior	750 horas	20€/hora	15.000€
TOTAL			20.160€

Tabla A.1: Costes de material y costes humanos

En la Tabla A.2 se pueden ver desglosadas cada una de las fases del proyecto, especificando tanto la duración de cada una de ellas como la dedicación en hora de los recursos detallados.

A.3. Presupuesto del proyecto

- Autor: Amelia Perales Egea
- Departamento: Teoría de la Señal y Comunicaciones (TSC)
- Descripción del proyecto: Aplicación de filtros de partículas distribuidos para la resolución del problema de estimación de la posición de un objeto en movimiento dentro de una red

Tarea	Duración [semanas]	Recursos [Ing./mes]	Total [horas]
Subtarea A.1	1	0,5	20
Subtarea A.2	1	0,5	20
Subtarea A.3	1	0,5	20
Subtarea A.4	3	0,5	60
Subtarea B.1	3	0,25	30
Subtarea B.2	3	0,25	30
Subtarea B.3	2	0,25	20
Subtarea B.4	3	0,5	60
Subtarea B.5	8	0,25	80
Subtarea B.6	6	0,25	60
Subtarea C.1	2	0,25	20
Subtarea C.2	9	0,5	180
Subtarea D.1	1	0,25	10
Subtarea D.2	13	0,5	260
Tarea E	1	0,75	30
TOTAL			900

Tabla A.2: Listado de las fases del proyecto y la dedicación en horas a cada una de ellas

de sensores, utilizando técnicas de fusión de datos para reducir el tráfico presente en la red y permitir así un aumento de la vida útil del sistema.

- Título: Fusión de Datos para Aplicaciones de Seguimiento Distribuido en Redes de Sensores.
- Duración: 12 meses.
- Tasas de costes indirectos: No se especifican.
- Presupuesto total del proyecto (€): 20.160€

Bibliografía

[1] Y. Sankarasubramaniam I. Akyildiz, W. Su and E. Cayirci, "Wireless sensor networks: A survey", March 2002.

[2] Feng Zhao and Leonidas Guibas, *Wireless Sensor Networks: An Information Processing Approach*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[3] *Fidelity and yield in a volcano monitoring sensor network*, 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2006). G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees and M. Welsh, November 2006.

[4] Petar M. Djuric, "Particle filters and their application to wireless communications", Tech. Rep., Stony Brook University, NY, USA.

[5] Sinem C. Ergen, "Zigbee/ieee 802.15.4 summary", Tech. Rep., EECS Berkeley, sep 2004.

[6] S. Pino-Povedano, R. Arroyo-Valles, and J. Cid-Sueiro, "Selective forwarding for energy-efficient target tracking in sensor networks", *Signal Processing*, 2013, Submitted - 2nd Review.

[7] Eduardo F. Nakamura, Antonio A. F. Loureiro, and Alejandro C. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications", *ACM Comput. Surv.*, vol. 39, no. 3, September 2007.

[8] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking", *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 425 –437, feb 2002.

[9] Theodore Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2001.

[10] A. J. Haug, "A Tutorial on Bayesian Estimation and Tracking Techniques Applicable to Nonlinear and Non-Gaussian Processes", Tech. Rep., 2005.

[11] Jianqui Zhang Perar M. Djuric, Jayesh H.Kotecha and Yufri Huang, "Particle filtering: A review of the theory and how it can be used for solving the problems in wireless communications", *IEEE Signal Processing Magazine*, September 2003.

[12] Anwer Bashi, Vesselin Jilkov, X. Rong Li, and Huimin Chen, "Distributed implementations of particle filters", in *In Proc. Sixth International Conference of Information Fusion*, 2003, pp. 1164–1171.

[13] Alexander T. Ihler, John W. Fisher Iii, and Alan S. Willsky, "Using sample-based representations under communications constraints", Tech. Rep., MIT, Laboratory for Information and Decision Systems, December 2004.

[14] Ramesh Rajagopalan and Pramod K. Varshney, "Data aggregation techniques in sensor networks: A survey", *Comm. Surveys and Tutorials, IEEE*, vol. 8, pp. 48–63, 2006.

[15] Stephanie Lindsey, Cauligi Raghavendra, and Krishna M. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics", *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 9, pp. 924–935, September 2002.